Stochastic Linear Bandits An Empirical Study

Students: DELAVANDE Julien & MEGDOUD Soël,

1 Problem 1

Here is the plot of the cumulative regret for linear epsilon greedy strategies for different epsilon in default settings with a 10000 finite horizon.

Linear Epsilon Greedy strategies

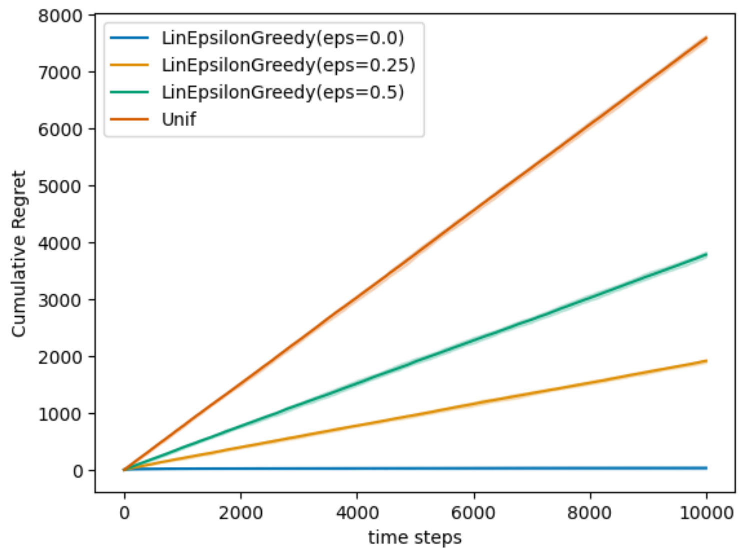

Figure 1: Cumulative regret for LinEpsilonGreedy and LinUniform policies

- The ϵ-Greedy policy outperforms the Uniform policy because while Uniform policy randomly selects actions without learning from rewards, ϵ-Greedy leverages observed rewards to estimate θ, choosing actions that maximize the estimated reward most of the time $(1-\epsilon)$ while occasionally exploring (ϵ).
- In $\epsilon = 0$, the policy becomes purely exploitative, selecting the action with the highest estimated reward based on $\hat{\theta}$. At the beginning of the game, there is *implicit exploration* due to the random errors in $\hat{\theta}$. Implicit exploration occurs because early estimations of $\hat{\theta}$ are unreliable, causing the agent to select diverse actions, effectively exploring the action space. This is effective since actions are uniformy sampled in ActionsGenerator() allowing to cover all the space. Once sufficient data is collected, $\hat{\theta}$ converges, and the agent consistently chooses the optimal arm.
- Implicit exploration in enough to converge, thus, explicit exploration $(\epsilon > 0)$ forces the policy to select suboptimal actions, increasing regret unnecessarily. This is reflected in the asymptotic gradient of the cumulative regret curve, which is proportional to ϵ, $\frac{\partial R(t)}{\partial t} \propto \epsilon$.

Complexity improvement

Regarding the complexity of the matrix inversion step, the basic algorithm at each time step:

1. Updates the covariance matrix $A = A + a_t a_t^\top$.
2. Computes the inverse of the updated covariance matrix $A^{-1} = \text{inv}(A)$.
3. Updates the parameter estimate $\hat{\theta} = A^{-1}b$.

This naive approach has a computational complexity of $O(d^3)$ for the matrix inversion step, which dominates the overall runtime, especially as the dimension d increases.

To improve the efficiency of the algorithm, we can use the Sherman-Morrison formula for incremental matrix updates. The key idea is to avoid recalculating the full inverse of A at each step. Instead the inverse of the covariance matrix A^{-1} is updated incrementally using:

$$A^{-1} \leftarrow A^{-1} - \frac{(A^{-1}a_t)(a_t^\top A^{-1})}{1 + a_t^\top A^{-1} a_t}$$

This reduces the complexity of the matrix update to $O(d^2)$.

The following table compares the execution times of both algorithms for various dimensions d. The experiments were conducted with $T = 10,000$ iterations.

Dimension (d)	**LinEpsilonGreedyFast (s)**	**LinEpsilonGreedy (s)**	**Time Gap (s)**
3	0.5367	0.5677	0.0310
10	0.5499	0.6128	0.0629
100	1.7780	50.8406	49.0625

Table 1: Runtime comparison for LinEpsilonGreedyFast and LinEpsilonGreedy algorithms across different dimensions.

Problem 2: LinUCB and LinTS

1. Implementation of LinUCB and LinTS

Both algorithms were implemented as follows:

- **LinUCB**: At each time step, selects action a_t maximizing the upper confidence bound:

$$a_t = \arg\max_a \left(a^\top \hat{\theta}_t + \alpha \sqrt{a^\top \Sigma_t a} \right).$$

- **LinTS**: Samples $\theta_t \sim \mathcal{N}(\hat{\theta}_t, \Sigma_t)$ and selects a_t maximizing the sampled reward:

$$a_t = \arg\max_a \ a^\top \theta_t.$$

2. Posterior at Time t

For Thompson Sampling, we assume a Gaussian prior:

$$\theta \sim \mathcal{N}(\mathbf{0}, \lambda I).$$

At time t, given actions $A_1, \ldots, A_t$ and rewards $Y_1, \ldots, Y_t$, the posterior distribution is:

$$\theta \sim \mathcal{N}(\hat{\theta}_t, \Sigma_t),$$

where:

$$\Sigma_t = \left(\lambda I + \frac{1}{\sigma^2} A_t^\top A_t\right)^{-1}, \quad \hat{\theta}_t = \frac{1}{\sigma^2} \Sigma_t A_t^\top Y_t.$$

3. Proposed Experiment

Goal: Determine the better algorithm among LinUCB, LinTS, Greedy and Epsilon-Greedy.
Setup:

- Environment: Linear bandit with $d = 30$, $K = 7$ actions, $T = 1000$ time steps, $\sigma^2 = 1$.
- Agents: LinUCB, LinTS, Epsilon-Greedy ($\epsilon = 0.1$), Greedy.
- Metric: Cumulative pseudo-regret averaged over 10 runs.

Results:

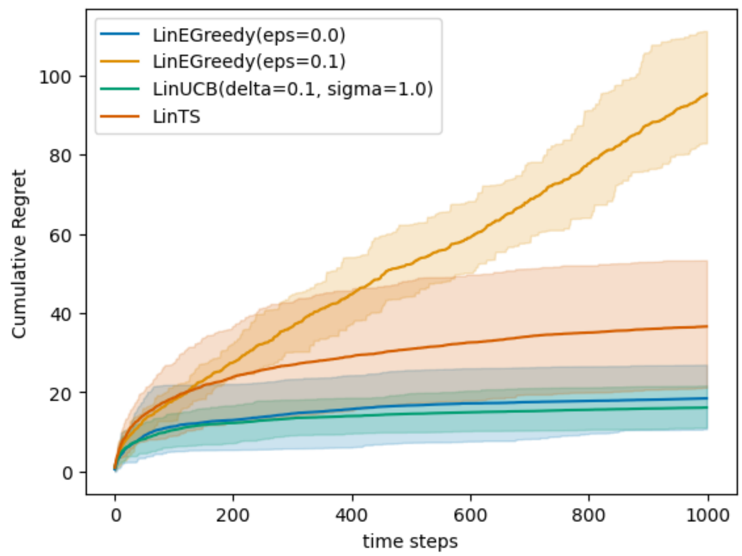

Figure 2: Mean Cumulative regret over time for Epsilon Greedy, Greedy, LinTS and LinUCB strategies

Conclusions:

- **LinUCB is the best:** LinUCB achieves the lowest cumulative regret due to its efficient confidence-bound-driven exploration, which balances exploration and exploitation optimally.
- **Greedy ($\epsilon = 0.0$) performs comparably:** Greedy has slightly higher regret than LinUCB but performs well because the environment favors exploitation, with actions well-distributed across the space.
- **LinTS is slightly worse than Greedy and LinUCB:** LinTS uses posterior sampling for exploration but does not outperform LinUCB or Greedy in this environment, likely due to the simplicity of the setup where deterministic exploration suffices.
- **Epsilon-Greedy ($\epsilon = 0.1$) is suboptimal:** Random exploration causes unnecessary regret, making it inefficient compared to the structured exploration of LinUCB and LinTS.
- **General takeaway:** In this environment, LinUCB performs best due to its structured exploration. However, Greedy is almost as effective, demonstrating that exploration is less critical in simple environments with well-distributed actions.