Adaptive Feature Selection For Sparse Linear Bandits

Experimental study on strategies for Online Feature Selection in High-Dimensional Bandit Settings

Martin Damyanov[1]
Supervisor: Julia Olkhovskaya[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Martin Damyanov
Final project course: CSE3000 Research Project
Thesis committee: Julia Olkhovskaya, Luciano Cavalcante Siebert

Abstract

The Multi-armed Bandit (MAB) is a classic problem in reinforcement learning that exemplifies the exploration-exploitation dilemma - deciding when to gather more information and when to act on current knowledge. In its sparse variant, the feature vectors often contain many irrelevant components, which is common in real-world scenarios and poses a significant efficiency bottleneck. Querying the entire vector is often unnecessary when only a small subset of it is informative. In this work, we introduce two novel algorithms for dynamic feature selection in sparse linear bandit settings. These algorithms adaptively select relevant features at each round, enabling more efficient learning and decision-making. Empirical results demonstrate that our methods consistently outperform standard approaches such as LinUCB in sparse environments.

1 Introduction

Stochastic linear bandits are a class of decision-making problems where a learning agent has to choose between a set of actions, each of which returns a different reward. The reward for every action is unknown to the agent and has to be estimated. This problem lies at the heart of reinforcement learning: should the agent exploit the information it already has, choosing the action with the best reward? Or should it explore actions it has less information about in an attempt to get better rewards in the long term?

Bouneffouf et al. outline some of the numerous cases in which the bandit framework has been applied in recent years [4]. Some of the real-life applications include treatment allocation procedures [12] [10], recommendation systems [22], and anomaly detection [9]. Bandit algorithms can also be used as the building blocks for better machine learning algorithms, for example, in hyperparameter optimization [17] and online feature selection [19].

In many of those real-world scenarios, decisions must be made based on high-dimensional data, where only a small subset of features truly influences the outcome. For instance, a system can recommend thousands of articles to a user, but only a few will genuinely capture their interest. This sparsity presents both a challenge and an opportunity: identifying and exploiting the relevant features can lead to more efficient learning and better performance. To solve this, there exist sparse bandit algorithms, which theoretically offer improved regret bounds [1]. However, they often do not explicitly incorporate feature selection mechanisms. These mechanisms could further enhance empirical performance, particularly in highly sparse regimes or when feature costs are constrained.

In this paper, we study how adaptive feature selection can be used to improve the performance of bandit algorithms in the sparse setting - that is, at every round, the algorithm has to pick a subset of the available features and decide on an action based on them alone. We focus specifically on the sparse case where most of the features do not influence the reward, making feature selection both relevant and beneficial. The question we want to answer is then: **How can we leverage feature selection strategies to design bandit algorithms that perform well in the sparse setting?** To address this, we introduce two algorithms and investigate various feature selection strategies through empirical evaluation.

[1] Formally defined in subsection 2.2

The first algorithm builds on a reduction from the bandit problem to an online linear regression problem, specifically leveraging the SquareCB approach [11]. We use two regression oracles to solve the resulting problem - sparse linear regression with adaptive feature selection [14] and Bayesian Linear Regression [21]. The second algorithm is inspired by FS-SCB [18] and operates using an ensemble of M models, each employing a different strategy to approximate the reward. These models serve as experts, each using a distinct subset of features, and their predictions are aggregated to guide action selection. The aim is for their combined output to approximate the reward of a model that has access to the optimal subset of features.

2 Problem Formulation

Throughout the paper, we use mathematical notation that might be unfamiliar to some readers, which we introduce in this section. Additionally, we give a formal description of contextual linear bandits.

2.1 General mathematical notation

The dot product between two vectors $a, b \in \mathbb{R}^n$ is indicated by $\langle a, b \rangle$. For any positive integer $n \in \mathbb{Z}^+$ we use $[n]$ to indicate the set $\{1, 2, ..., n\}$. By $a \circ b$ we denote the Hadamard product of two vectors, also known as the element-wise product. A vector is said to be s-sparse if it has s non-zero entries. x_i is used to denote the i-th entry of the vector x.

For a subset $S \subseteq [n]$ we define $x(S)$ to be the projection of x such that

$$x(S)_i = \begin{cases} x_i & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}$$

Moreover, we define the support of a vector $x \in \mathbb{R}^n$, $\text{supp}(x)$, to be the set of all of its non-zero coordinates or $\text{supp}(x) = \{i \mid i \in [n] \wedge x_i \neq 0\}$.

2.2 Contextual Linear Bandits

A linear bandit problem can be defined as a game over T rounds where $T \in \mathbb{Z}^+$ is called the *horizon*. At every round $t \in [T]$, an agent is given a *context* $c_t \in \mathcal{C}$. It then plays an *action* $a_t \in \mathcal{A}$ where $\mathcal{A} = [K]$ is the set of all possible actions and receives a *reward* $r_t(a_t, c_t)$ based on both the context and the action that was picked. Here, K is the number of actions available to the agent. To generate the reward, the environment picks a parameter $\theta^* \in \mathbb{R}^d$ unknown to the agent. Note that θ^* does not change for the duration of the game. Our estimation of θ^* does, and we denote it by $\hat{\theta}$. The reward in round t for action a_t would then be $r_t(a_t, c_t) = \langle \phi(a_t, c_t), \theta^* \rangle + \eta_t$ where $\phi : \mathcal{A} \times \mathcal{C} \to \mathbb{R}^d$ is a known function that transforms the action and the context into a *feature vector*. ϕ is also called the *feature map* of the model. η_t is σ-sub-gaussian noise. A random variable is called σ-sub-gaussian if there is some $C > 0$ such that for every $a \geq 0$

$$\mathbb{P}(|X| \geq a) \leq 2 \exp\left(-a^2/C^2\right)$$

The goal of the agent is to maximize the total reward, or equivalently, to minimize the *regret*, which can be defined as the difference in total reward between the best-performing strategy that has full information about θ^*, and the total reward of our strategy. Mathematically, this can be expressed as follows,

$$\mathcal{R}(T, \theta^*) = \sum_{t=1}^{T} \Big(\langle \phi(a_t^*, c_t), \theta^* \rangle - \langle \phi(a_t, c_t), \theta^* \rangle \Big)$$

With a_t we indicate that action a was chosen at round t by our agent. a_t^* is the optimal action at round t or $a_t^* = \text{argmax}_{a_t \in \mathcal{A}} \langle \phi(a_t, c_t), \theta^* \rangle$. A bandit is called sparse if θ^* is a sparse vector, i.e. it has many zero entries. Sparsity influences the problem because only a small subset of its coordinates of θ^* will influence the reward. Learning the structure of θ^* can therefore be beneficial for improving the regret bounds.

3 Related work

Stochastic linear bandit algorithms are subject to a $\Omega\big(d\sqrt{T}\big)$ regret lower bound regardless of how many features are informative [8]. This implies that regret scales linearly with the dimension of θ^*, d, even if only a few features matter. To address this, sparse linear bandits assume that the true parameter θ^* is s-sparse, where $s \ll d$. In this setting, the lower bound of the regret improves to $\Omega\big(\sqrt{sdT}\big)$, which matches the standard linear bandit regret when $s = d$.

Abbasi-Yadkori et al. [1] achieve this bound with a variant of UCB tailored to the sparse setting. Hao et al. [3] achieve the optimal regret bounds in the data-poor regime where $d \gg T$, showing their algorithm achieves $\Theta(\sqrt{sdT})$ regret when $d \leq T^{1/3}s^{2/3}$, and degrades to $\Omega(T^{2/3})$ when $d > T^{1/3}s^{2/3}$. Assuming a different model for the noise and the action set one can achieve an algorithm with regret bounds that does not scale with d. Carpentier and Munos [6] achieve a $\mathcal{O}\big(s\sqrt{T}\big)$ regret bound by assuming that the action set is the unit ball (vectors with length ≤ 1) and that there is noise in the features, or $r_t = \langle \phi(a_t, c_t), \theta^* + \eta_t \rangle$. Lattimore and Szepesvári [16] achieve the same regret bound under the assumption that the action set is a hypercube $\mathcal{A} = [-1, 1]^d$ and the noise is bounded in $[-1, 1]$.

An example of a field that could benefit from feature selection in bandits is that of distributed bandit algorithms. Several efforts have addressed the problem of minimizing the communication costs between distributed bandits. Those communication costs are often highly dependent on the dimensionality of the data. Hanna et al. [13] establish a lower bound on the number of bits required for agents to communicate rewards to a central learner in a distributed setup. In their model, agents do not have memory and receive only the action to play, responding with a bit sequence encoding the observed reward. While they show promising results, the setting is restricted since there is only one central learner. Wang et al. [20] examine a multi-agent setting where agents operate in a shared environment and coordinate to minimize regret. They provide a communication complexity bound of $\mathcal{O}(M^{1.5}d^3)$ when the set of available actions changes over time, where M is the number of agents. Since the dimensionality is cubed, these communication bounds can be vastly improved upon with an appropriate feature selection strategy, especially in the sparse setting. However, to the best of our knowledge, no previous work has explicitly explored feature selection in the context

of sparse bandit problems.

4 Bayesian Feature Selection

For the rest of the paper, we make extensive use of a strategy for feature selection called Bayesian Feature Selection. We describe it here and introduce the relevant notation.

Bayesian Feature Selection is a version of Bayesian Linear Regression [2]. It leverages the posterior probabilities of the weights to calculate a distribution over the features and determine which ones are important. It works by introducing a new random variable γ_k such that the prior for $\hat{\theta}_k$ can be written as

$$\begin{aligned} \hat{\theta}_k \mid (\gamma_k = 0) &\sim \mathcal{N}(0, \tau^2) \\ \hat{\theta}_k \mid (\gamma_k = 1) &\sim \mathcal{N}(0, c\tau^2) \end{aligned} \tag{1}$$

where $\tau \ll 1$ is a very small positive constant and $c \gg 1$ is a large positive constant. This is to make the density of $\hat{\theta}_k$ shrink to 0 when $\gamma_k = 0$. The goal of the algorithm is to estimate $\mathbb{P}(\gamma \mid X, Y)$. We can do this by numerically calculating the value of the following integral, which we do using Gibbs sampling [15].

$$\mathbb{P}(\gamma \mid X, Y) = \int \mathbb{P}(\gamma, \hat{\theta} \mid X, Y) d\hat{\theta} \tag{2}$$

Note that a regular numerical integration scheme will fail due to the high dimensionality of the integral, which scales with the dimension of θ^*. A full description of the method can be found in [21].

The probability $\mathbb{P}(\gamma_k = 1 \mid X, Y)$ shows how likely it is for a feature to be informative, or to correspond to a non-zero entry of θ^*. We can therefore do feature selection by sampling from that distribution.

5 Adaptive feature selection with a regression oracle

In this section, we explore an approach to solving bandit problems using a linear regression oracle, SquareCB [11]. We use two different oracles to perform adaptive feature selection, mainly Bayesian Feature Selection and FSSLR [14].

SquareCB

SquareCB [11] is an optimal and universal reduction from bandits to online regression. It allows us to transform an oracle for online linear regression into an algorithm for linear bandits. The reduction can be treated as a black box, meaning that it will take any regression oracle and use it to solve the bandit problem. Under the assumption of realizability, i.e. that there exists a regressor f in the class of regressors we have chosen such that $f(\phi(a_t, c_t)) = E[r_t \mid (a_t, c_t)]$ we know that the regret can be bounded as $\mathcal{O}\big(\sqrt{KT \cdot \mathcal{R}_{sq}(T)}\big)$

where $\mathcal{R}_{sq}$ is the regret of the regression oracle.

It has been shown that the regret of regularized linear regression, such as Ridge, scales logarithmically with T but linearly with d, the dimension of the data [7]. This means that while it can be a good choice for a regression oracle if the data is low-dimensional, its performance will deteriorate as the dimension increases. Therefore, for the sparse setting, we would benefit from an oracle whose regret scales better with d. Our first choice for such a regression oracle is the Bayesian Regression with Feature Selection algorithm mentioned in section 4. The second choice is a special type of sparse online linear regression, which we call FSSLR [14] (Feature Selection Sparse Linear Regression).

Bayesian Regression with Feature Selection

The approximation of the integral in Equation 2 involves computing the posterior distribution of $\hat{\theta} \sim \mathbb{P}(\hat{\theta} \mid \gamma, X, y)$ which gives us an approximation of the real value of θ^*. Here, γ is a binary vector that indicates which features have been selected, with ones at positions corresponding to the chosen features and zeroes elsewhere. The predicted reward is then given by $r_t = \langle \phi(a_t, c_t) \circ \gamma, \hat{\theta} \rangle$.

Since we use a Gaussian prior as described in Equation 1, sampling from the posterior simplifies to sampling from a Gaussian distribution whose mean is given by the closed-form solution of the Ridge estimator - a standard result in Bayesian linear regression [2]. This implies that the regret of this regressor will scale linearly with the dimension of θ^*, d. Formal analysis on whether incorporating feature selection alters this scaling behavior remains an open problem. We explore it through empirical analysis, which shows an improvement.

FSSLR

FSSLR is a variant of sparse online linear regression that performs random feature selection at every round. It then uses a modified version of the Dantzig selector [5] to find an optimal choice for the weights of the regression. For any $\delta > 0$ its regret can be bounded as

$$\mathcal{R}(T) = \mathcal{O}\left(s^2 \log\left(\frac{d}{\delta}\right) \left(\frac{d}{k}\right)^3 \log(T) \right) \tag{3}$$

with probability at least $1 - \delta$. Here k is used to indicate how many features are selected every round. Equation 3 shows that when the data is sufficiently sparse, the algorithm can achieve more efficient regret bounds than Ridge.

At round t, FSSLR receives a feature vector x_t. It generates a random set $S_t \subseteq [d]$ with $|S_t| = k$, which is used to generate an unbiased estimate of x_t by $\hat{x}_t = \frac{d}{k} x_t(S_t)$. $\frac{d}{k}$ is debiasing factor to ensure that $\mathbb{E}[\hat{x}_t] = x_t$. This is appended to the feature matrix $\hat{X}_{t-1}$ and is used in the calculation of the weights of the linear regression. A major drawback of this approach is that since we are randomly generating S_t at every round, if k is too close to s, the probability that any of the significant features of x_t are captured in $\hat{x}_t$ decreases drastically. Since θ^* is the sparse weight vector that the linear regression is trying to approximate, we can write down the probability of capturing all important features as

$$\mathbb{P}(\text{supp}(\theta^*) \subseteq S_t) = \frac{C_{k-s}^{d-s}}{C_k^d} \tag{4}$$

where C_k^d is the number of all possible combinations of size k that we can take from a set of size d. That means that when k decreases, the algorithm will need significantly more rounds for $\hat{X}_t$ to be a good enough estimator of the actual feature matrix, resulting in significantly worse regret bounds. This is also reflected in the factor of d/k in the regret, which is cubed. On the other hand, when the sparsity increases, we see an improvement in the performance because the probability of randomly finding the full support of θ^* is larger.

Another drawback of this approach is that FSSLR assumes that its data follows the Restricted Isometry Property (RIP) [5]. When dealing with real data, this will rarely hold, so for our analysis, we did not explicitly ensure that RIP holds. This means that the theoretical guarantees that the original paper provides might not necessarily hold in this setup.

6 Adaptive feature selection with model selection

In this section, we explore another strategy for feature selection based on FS-SCB [18]. It works by defining multiple models that it treats as experts. Their predictions are then averaged using an algorithm for aggregating expert advice. We also introduce two strategies for creating the models. The first one is random, the second one observes the structure in a short warm-up period and then performs Bayesian Feature Selection.

FS-SCB

FS-SCB [18] is an algorithm designed to identify the correct feature map from a set $\Phi = \{\phi^i\}_{i=1}^M$ of M distinct feature maps (or models), assuming at least one of them lies in the linear span of the true reward function i.e. there is some $i \in [M]$ such that $\mathbb{E}[r_t] = \langle \phi^i(a_t, c_t), \theta^* \rangle$. We propose a modification to FS-SCB to enable adaptive feature selection, where each feature map is defined as a projection of the original d-dimensional feature vector onto a k-dimensional space with $d > k$. This way we are effectively simulating feature selection where k is the number of features that we want to select. Unlike the original FS-SCB, which relies on the presence of a correctly specified feature map within Φ, our approach deliberately excludes the true feature map from Φ. Instead, we aim to discover a reduced representation that still captures the essential structure of the reward function.

The theoretical guarantees of FS-SCB largely depend on the fact that it can detect misspecified maps and remove them from Φ. That being said, this is not as effective in our setting, since the true map might not be in Φ. Given that we are assuming a sparse θ^*, it is still possible to generate a feature map that retains all relevant information. This feature map would retain all coordinates in the feature vector corresponding to a non-sparse entry in θ^*. However, since we do not know θ^*, we cannot guarantee that. Furthermore, because most feature maps are deliberately constructed to closely approximate the true map, it becomes challenging to determine whether any are truly outside the linear span of the reward function or just a good approximation. The limitation of this approach is that the algorithm's elimination threshold is rarely exceeded if the maps are similar enough to the true one, which leads to no models being removed. This limitation arises not from a flaw in the algorithm itself but from the nature of our map construction. Importantly, this does not compromise the algorithm's theoretical guarantees, as Φ was designed to include maps that are structurally close to the true one. Model elimination is therefore omitted from further

discussions since it has no impact on the performance of the algorithm.

As mentioned above, the best candidate for a feature map would be the one that projects only the set of coordinates in the feature vector that correspond to non-sparse entries in θ^*. To get as close as possible to that, we have developed multiple strategies for feature map selection, which we describe now.

Random model selection

With *random model selection*, we uniformly select a random subset $S^i \subseteq [d]$ for each feature map with $|S^i| = k$. The feature map is then defined as $\phi^i(x) = x(S^i)$ where $x(S^i)$ is k-sparse. The probability of an individual feature map being in the span of the reward is shown in Equation 4. Since we have M distinct feature maps and want at least one of them to be true, the probability becomes

$$\mathbb{P}\Big(\bigvee_{i=1}^{M} \text{supp}(\theta^*) \subseteq S_i\Big) = 1 - \left(1 - \frac{C_{k-s}^{d-s}}{C_k^d}\right)^M \tag{5}$$

Notice that since in this case we treat each of the M models as an expert whose predictions we aggregate, Equation 5 is not necessarily indicative of the performance of the algorithm, although it can show us a theoretical region in the hyperparameter space where the algorithm is sure to perform well (i.e. the one where the probability will be high). From it we see that increasing M will increase our chance of capturing the true feature map exponentially. That being said, increasing M results in a huge runtime overhead. The only other parameter we can vary is k, the number of features selected every round. Ideally, we would like k to be as close as possible to s if we want to find the optimal feature set $S_i = \text{supp}(\theta^*)$. However, when k approaches s, the probability of finding it decreases significantly. To put it into perspective, if we have $d = 100$, $s = 5$, and $k = 20$, we would need more than 85000 randomly generated feature maps if we want $\geq 95\%$ chance of finding the true one. The runtime-accuracy tradeoff that this selection strategy offers us calls for a more sophisticated method of selecting the models, which has a better probability of finding feature maps close to the real one.

Informed model selection

Many meaningful alternatives to random feature selection exist. However, all of them require that the algorithm observes some of the data and makes an informed decision based on what it has seen. This will come at the cost that the algorithm needs to sacrifice the first t_w rounds just to observe the data so that it can make a meaningful decision after.

Our chosen approach for feature selection is the Bayesian Feature Selection algorithm described in section 4. We can incorporate it into FS-SCB as follows: for the first t_w rounds, we observe the data by using a single feature map that contains the full set of features. At round t_w+1 we calculate the distribution $\mathbb{P}(\gamma \mid X, Y)$ and we sample it M times to construct our feature maps as follows

$$\phi^i(a_t, c_t) = \gamma_i \circ \phi(a_t, c_t)$$

where ϕ is the true feature map and $\gamma_i \sim \mathbb{P}(\gamma \mid X, Y)$.

7 Experimental Setup and Results

In this section, we explain the experimental setup in which we empirically test the performance of the algorithms, explain how they are assessed, and finally give the results. All results are accompanied by a discussion.

Environment

Testing of the algorithms happened in a synthetic environment. For action a_i we generate a normal distribution with mean $\mu_{a_i} \sim U(-2, 2)$ and variance $\sigma^2 \sim U(0, 1)$ where U represents the uniform distribution. The feature vector for action a_i is then $\phi(a_i) \sim \mathcal{N}(\mu_{a_i}, \sigma^2)$. Note that the context is not directly used in the calculation of the feature vector, but instead, we have a separate context behind each action, namely the normal distributions that we create in the beginning. This does not change the problem statement as we generate the feature vector directly. Similar approaches have been used in [18]. The dimensionality of our environment is $d = 100$. Unless otherwise specified the number of actions is $K = 100$ and sparsity of θ^* is $s = 5$. The noise was sampled from $\mathcal{N}(0, 0.01)$.

Assessment of performance

We assess the performance of the algorithms based on plots of their cumulative regret. A completely random algorithm will have a linearly increasing regret, so we say that an algorithm is learning if it has sub-linear cumulative regret. Ideally, we want the cumulative regret to grow as slowly as possible, where a cumulative regret of 0 can be attained only by an algorithm with full information about the hidden parameters from the start. Each plot shows how the algorithm performed over multiple trials (5 or 10, depending on the variance and runtime of the algorithms). In all plots, we indicate the variance of the algorithms between runs by coloring a region of one standard deviation around the mean.

Performance is assessed relative to LinUCB, which we use as a baseline. To get a more accurate idea of how our algorithms compare, we also include a version of LinUCB that has access to a limited subset of the features at every round. These subsets are selected uniformly and at random.

Performance of SquareCB with FSSLR

We evaluate SquareCB using FSSLR as the regression oracle to test its ability to adapt to sparsity under different configurations. The performance of this algorithm is highly dependent on the hyperparameter k, which determines the number of features selected at each round.

As shown in Figure 1, the algorithm achieves sublinear cumulative regret only when k is sufficiently large compared to the sparsity s of the hidden parameter θ^*. When k is close to s, the probability of capturing the full support of θ^* in any given round becomes small, leading to a poor estimate of the full feature matrix X and nearly linear regret growth. In contrast, increasing k improves the coverage of the relevant feature set, resulting in significantly better performance.

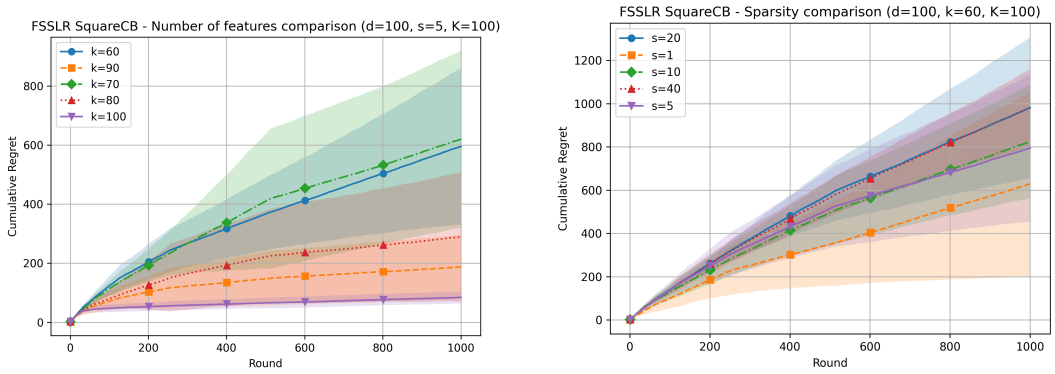

Figure 1: SquareCB with FSSLR for different values of the selected features and the sparsity. Shows that decreasing the number of features selected increases the regret, and making the environment sparser decreases the regret.

SquareCB with FSSLR is specifically designed for sparse data and consistently outperforms LinUCB in such settings if it has access to a sufficiently large subset of the features. The results, shown in Figure 6, demonstrate that SquareCB with FSSLR can achieve significantly lower regret for high values of k. However, this trade-off is of limited practical value: reducing k inevitably compromises accuracy while offering minimal gains in runtime efficiency. That said, it may still be useful in scenarios where memory constraints are critical, as limiting k reduces the number of features that need to be queried and stored.

Some readers might then wonder why we still use random feature selection with FSSLR if it does not perform well for small values of k. Why not use a smarter strategy for feature selection, for example, looking at the estimate of the weights in $\hat{\theta}$ and picking the features corresponding to the highest weights in $\hat{\theta}$? Assume that $S_t \subset [d]$ is the set with the indices with the highest absolute values in $\hat{\theta}$ at round t. The expectation of a single element of the projected matrix $\hat{X} = X(S_t)$ would then be

$$\mathbb{E}[\hat{X}_{ij}] = \mathbb{P}(\hat{X}_{ij} \neq 0 \mid i \in S_i)\mathbb{P}(j \in S_i)\hat{X}_{ij} + \mathbb{P}(\hat{X}_{ij} = 0 \mid j \notin S_i)\mathbb{P}(j \notin S_i)\hat{X}_{ij}$$

The probability $\mathbb{P}(j \in S_i)$ is the probability that the Dantzig selector picks a feature j at round i, which at least to our knowledge, cannot be computed. This means that each entry of the matrix will be inherently biased by a factor of $\mathbb{P}(j \in S_i)$. Our experiments show that this reduces the quality of the predictions significantly.

Performance of SquareCB with Bayesian Feature Selection

SquareCB with Bayesian Feature Selection shows much better performance than with FSSLR, as shown in Figure 6.

A critical implementation detail concerns how we store the feature matrix X. If X contains only the selected (projected) features rather than the full feature vectors, performance degrades significantly. However, storing the full matrix undermines the goal of feature selection, as predictions are effectively made using all features. Despite the drop in performance,

we choose to store only the selected features and use this version throughout the paper. Note that FSSLR also stores the projected matrix. The difference between both implementations for Bayesian Feature Selection is shown in Figure 2. To debias the projected matrix, we would need to divide it by $\mathbb{P}(\gamma = 1 \mid X, Y)$. However, this probability converges rapidly, making debiasing attempts ineffective and leading to larger errors due to numerical and approximation issues.

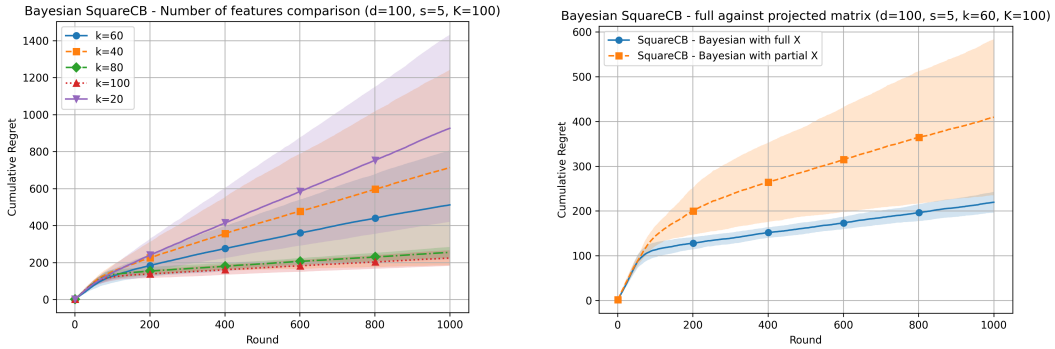

Figure 2: SquareCB with Bayesian Feature Selection. The left plot shows how the regret scales with changes in k. Access to a bigger subset of the features always leads to better regret bounds. On the right, we see the difference between Bayesian SquareCB with full and projected matrix. Storing the full matrix leads to better regret bounds.

Performance of FS-SCB

FS-SCB consistently achieves sublinear regret across a wide range of hyperparameters. Its ability to aggregate predictions from different models makes it more robust to suboptimal feature subsets. The Bayesian model selection strategy computed after an initial warm-up phase significantly outperforms random selection by increasing the probability that relevant features are consistently included in the selected subsets.

As illustrated in Figure 4, increasing M improves the approximation of the true reward function, though the gains diminish rapidly beyond a certain point, as only one of those M models has to be a good enough approximation of the true feature map. Figure 3 illustrates how both selection strategies depend on the parameter k.

Interestingly, Bayesian feature selection with FS-SCB performs better when fewer features are selected, up to a certain point. This is likely due to the estimates of the zero entries in $\hat{\theta}$ being noisy. By selecting features that are more likely to correspond to non-zero entries in θ^*, the resulting reward estimates become significantly less noisy. This is the only algorithm we present that exhibits improved performance with a reduced feature set. In contrast, the other algorithms generally degrade in performance under similar constraints. We attribute this to the use of an ensemble of M models, which can collectively compensate for missing features. If one model overlooks an important feature, others in the ensemble are likely to capture it.

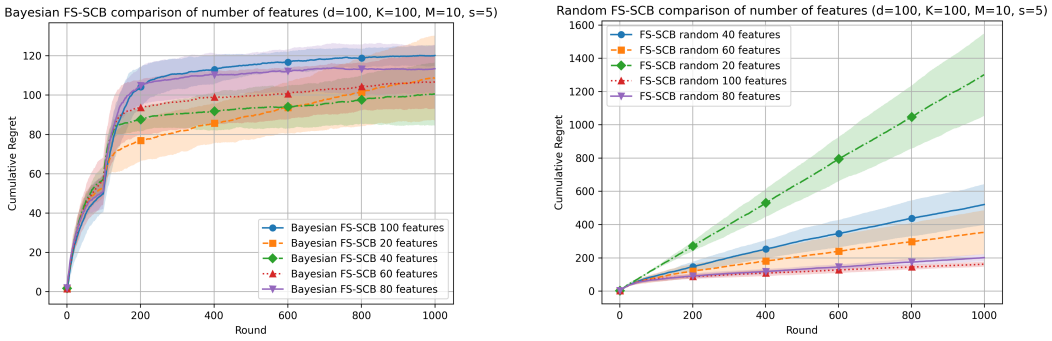

Figure 3: FS-SCB performance with varying k, using 10 fixed models. Informed selection strategies (left) show better regret bounds with smaller feature subsets (up to a certain point), while random selection's (right) regret degrades significantly as k decreases.

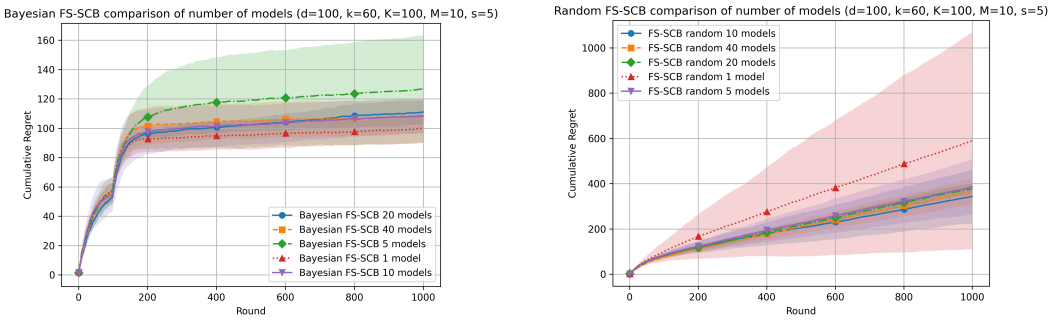

Figure 4: FS-SCB with a varying number of models. Shows how increasing the number of models has diminishing returns in the performance, which is especially evident with Bayesian feature selection.

Discussion on the relative performance of all algorithms

Generally, all strategies that made feature selection randomly performed worse than the ones that did it in an informed way. For high values of k, random strategies still outperform the baseline algorithm LinUCB, but for small values of k their performance degrades to nearly linear, except for FS-SCB with Bayesian Feature Selection (see Figure 6). This is visualized in Figure 5, where we show the probability from Equation 5. With yellow, we indicate regions where the probability of capturing the feature set is near one. This means that in those regions, the algorithms will perform well with a very high probability. Note that, especially for FS-SCB, a dark region does not necessarily mean poor performance. However, there is a significant correlation between how well an algorithm performs and the probability of it finding the correct feature subset.

The plot on the right has $M = 1$, which reduces to Equation 4. This shows that aggregation of model predictions can improve regret bounds significantly by increasing the probability of the correct feature map being selected. On the other hand, when relying on only one model (as is the case with SquareCB with FSSLR), performance is highly sensitive to k, as indicated by the size of the low probability regions when $M = 1$ in Figure 5. Both contour plots are only for strategies with random feature selection.

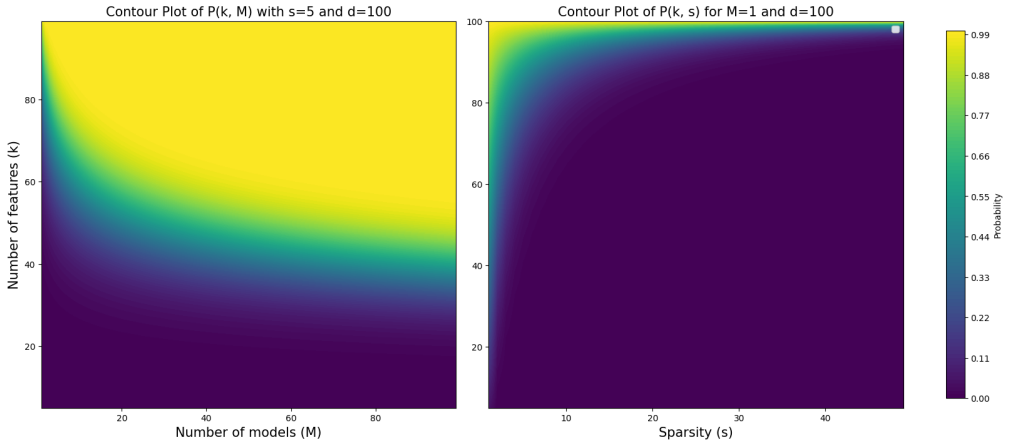

Figure 5: Contour plot of Equation 5 showing the dependence on M and k (left) and k and s (right). Dark blue indicates near-zero probability regions, and yellow indicates near-one probability regions.

Informed feature selection strategies, which we implemented with Bayesian Feature Selection, showed a much better performance for all tested values of k. They consistently outperform not only both baselines but also their random counterparts. FS-SCB with Bayesian Feature Selection shows the best performance across all tested configurations, as shown in Figure 6. In contrast, random selection strategies rapidly degrade in performance with smaller feature sets.

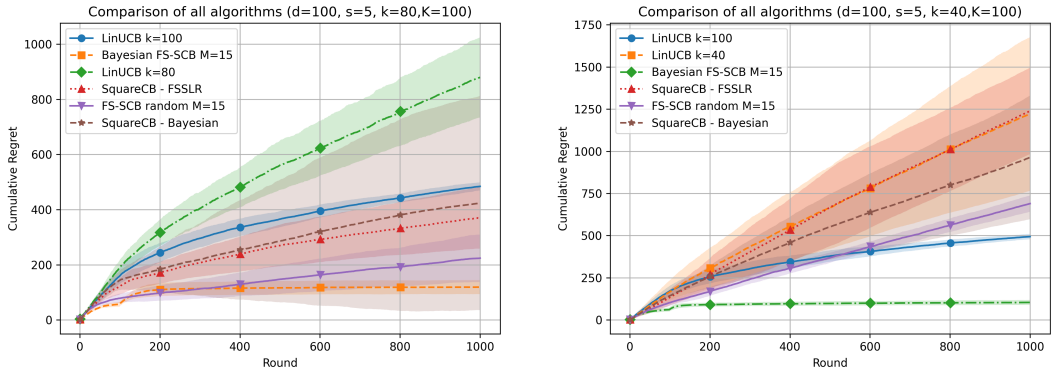

Figure 6: Full comparison of all algorithms under different values of k. Random feature selection works well for high values of k, but its performance degrades quickly as k decreases.

8 Responsible Research

Ethical implications of our work

The bandit framework has many real-world applications, and with that comes the responsibility to consider the ethical implications of research in this area. Since our algorithms were trained and evaluated solely on synthetically generated data, issues such as data privacy and the ethical handling of real user data were not a concern in our work.

Nevertheless, it is essential to reflect on how these algorithms might be applied in practice. Since bandit algorithms are inherently online and adaptive, their initial performance is often suboptimal. In practical settings, this limitation can be mitigated through pretraining on existing datasets, where available, to improve early-stage performance.

The ethical considerations associated with bandit algorithms can differ significantly across domains. In high-stakes areas like healthcare, for instance, using bandit algorithms to optimize treatment strategies must be approached with caution. It is critical not to over-rely on automated decision-making, as clinicians often possess nuanced insights that cannot be captured by data alone. In contrast, in recommender systems, bandit algorithms might be employed to manipulate user behavior, such as encouraging engagement with advertisements or content that users would not typically seek out, raising concerns about autonomy and informed consent.

Ultimately, the ethical use of these algorithms depends largely on those who implement them. While our role is primarily in developing the underlying methods, we believe it is important to highlight their potential risks. By doing so, we aim to equip practitioners and researchers with the awareness needed to make responsible choices in deploying these algorithms.

Reproducibility of research

We took great care to ensure that all results in this paper are fully reproducible. To this end a framework was developed that enables saving and loading experiment configurations using JSON files. These configuration files include all relevant hyperparameters for both the environments and the algorithms, allowing experiments to be rerun with exact precision. To account for the inherent randomness in our setup, each algorithm was executed multiple times, and the results were averaged to obtain consistent and reliable performance metrics.

The configurations for all experiments in this paper and the code have been made public and can be accessed at our GitHub [2]. Additionally, the exact code used to generate all plots in this paper is provided in accompanying Python notebooks within the repository.

9 Conclusion

Adaptive feature selection can be leveraged in bandit problems to effectively exploit sparsity. With well-tuned hyperparameters, the algorithms presented in this paper achieve notable improvements in regret bounds over the chosen baseline, LinUCB. That said, each variant of SquareCB and FS-SCB exhibits distinct strengths and limitations, as summarized in Table 1. While feature selection can enhance regret bounds in sparse settings, it offers minimal impact on the overall runtime of the algorithms. Moreover, these algorithms were not developed with theoretical optimality as the primary goal; instead, we prioritized practical performance.

Table 1: Comparison of Contextual Bandit Algorithms with Adaptive Feature Selection

Algorithm	**Strengths**	**Weaknesses**	**Best When**
SquareCB + FSSLR	Fast execution; strong theoretical guarantees under RIP	Highly sensitive to the size of feature sets; assumes RIP (often invalid)	Feature relevance is unknown, approximate results are acceptable, and k can be large; runtime is a priority
SquareCB + Bayesian Feature Selection	Robust to choice of k; consistently better empirical performance in sparse settings	Computationally heavier than FSSLR; depends on the quality of posterior estimation	Prior observations can be used to inform feature relevance
FS-SCB + Bayesian Model Selection	High accuracy; robust across different k; consistently best performance	Requires a warm-up period; significant runtime and memory overhead	A short initialization phase is acceptable; very low values for k are a priority

Future work could explore establishing theoretical regret bounds for the algorithms introduced in this paper that depend on various feature selection strategies, or deriving strategies

[2] `https://github.com/Martin092/ResearchProject`

with a better theoretical performance (or one that is easier to prove). Additionally, it would be valuable to conduct an empirical and theoretical comparison with existing bandit algorithms specifically designed for high-dimensional sparse settings, such as the approach proposed in [3] or in [1]. Such comparisons could help clarify the relative advantages and limitations of adaptive feature selection in various sparse regimes.

References

[1] Yasin Abbasi-Yadkori, David Pal, and Csaba Szepesvari. Online-to-Confidence-Set Conversions and Application to Sparse Stochastic Bandits. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, pages 1–9. PMLR, March 2012. ISSN: 1938-7228.

[2] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006.

[3] Mengdi Wang Botao Hao, Tor Lattimore. High-dimensional sparse linear bandits. *NIPS: Neural Information Processing Systems*, 2021.

[4] Djallel Bouneffouf and Irina Rish. A Survey on Practical Applications of Multi-Armed and Contextual Bandits, April 2019. arXiv:1904.10040 [cs].

[5] Emmanuel Candes and Terence Tao. The Dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 35(6), December 2007. arXiv:math/0506081.

[6] Alexandra Carpentier and Remi Munos. Bandit Theory meets Compressed Sensing for high dimensional Stochastic Linear Bandit. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, pages 190–198. PMLR, March 2012. ISSN: 1938-7228.

[7] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 1 edition, March 2006.

[8] Yan Dai, Ruosong Wang, and Simon S. Du. Variance-Aware Sparse Linear Bandits, February 2023. arXiv:2205.13450 [cs].

[9] Kaize Ding, Jundong Li, and Huan Liu. Interactive Anomaly Detection on Attributed Networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 357–365, New York, NY, USA, January 2019. Association for Computing Machinery.

[10] Audrey Durand, Charis Achilleos, Demetris Iacovides, Katerina Strati, Georgios D. Mitsis, and Joelle Pineau. Contextual Bandits for Adapting Treatment in a Mouse Model of de Novo Carcinogenesis. In *Proceedings of the 3rd Machine Learning for Healthcare Conference*, pages 67–82. PMLR, November 2018. ISSN: 2640-3498.

[11] Dylan J. Foster and Alexander Rakhlin. Beyond UCB: Optimal and Efficient Contextual Bandits with Regression Oracles, June 2020. arXiv:2002.04926 [cs].

[12] Mohsen Bayati Hamsa Bastani. Online decision-making with high-dimensional covariates. *Operations Research*, 2015.

[13] Osama A. Hanna, Lin Yang, and Christina Fragouli. Solving Multi-Arm Bandit Using a Few Bits of Communication. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, pages 11215–11236. PMLR, May 2022. ISSN: 2640-3498.

[14] Satyen Kale, Zohar S. Karnin, Tengyuan Liang, and Dávid Pál. Adaptive feature selection: Computationally efficient online sparse linear regression under RIP. *CoRR*, abs/1706.04690, 2017.

[15] Wonyoung Kim, Sungwoo Park, Garud Iyengar, Assaf Zeevi, and Min-hwan Oh. Linear bandits with partially observable features, Feb 2025.

[16] Tor Lattimore and Csaba SzepesvÃ¡ri. *Bandit Algorithms*. Cambridge University Press, 1 edition, July 2020.

[17] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization, June 2018. arXiv:1603.06560 [cs].

[18] Ahmadreza Moradipari, Berkay Turan, Yasin Abbasi-Yadkori, Mahnoosh Alizadeh, and Mohammad Ghavamzadeh. Feature and parameter selection in stochastic linear bandits, 2022.

[19] Jialei Wang, Peilin Zhao, Steven C.H. Hoi, and Rong Jin. Online Feature Selection and Its Applications. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):698–710, March 2014.

[20] Yuanhao Wang, Jiachen Hu, Xiaoyu Chen, and Liwei Wang. Distributed Bandit Learning: Near-Optimal Regret with Efficient Communication, May 2019. arXiv:1904.06309 [cs].

[21] Eric P Xing and Fan Guo. Bayesian Feature Selection.

[22] Qian Zhou, XiaoFang Zhang, Jin Xu, and Bin Liang. Large-Scale Bandit Approaches for Recommender Systems. In Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy, editors, *Neural Information Processing*, pages 811–821, Cham, 2017. Springer International Publishing.

Appendix A Use of Generative Artificial Intelligence in this work

AI (free version of ChatGPT [3]) was used solely for improving the writing style and clarity of the paper. Typical prompts were along the lines of: "Rewrite this paragraph and improve the wording where appropriate, without altering the underlying facts.". The given feedback was then carefully assessed, and only relevant parts were applied. No AI assistance was used for coding or idea generation.

[3] `https://chatgpt.com/`