Spectral Bandits for Smooth Graph Functions

Michal Valko
INRIA Lille - Nord Europe, SequeL team, 40 avenue Halley 59650, Villeneuve d’Ascq, France

Rémi Munos
INRIA Lille - Nord Europe, SequeL team, France; Microsoft Research New England, Cambridge, MA, USA

Branislav Kveton
Technicolor Research Center, 735 Emerson St, Palo Alto, CA 94301, USA

Tomáš Kocák
INRIA Lille - Nord Europe, SequeL team, 40 avenue Halley 59650, Villeneuve d’Ascq, France

Abstract

Smooth functions on graphs have wide applications in manifold and semi-supervised learning. In this paper, we study a bandit problem where the payoffs of arms are smooth on a graph. This framework is suitable for solving online learning problems that involve graphs, such as content-based recommendation. In this problem, each item we can recommend is a node and its expected rating is similar to its neighbors. The goal is to recommend items that have high expected ratings. We aim for the algorithms where the cumulative regret with respect to the optimal policy would not scale poorly with the number of nodes. In particular, we introduce the notion of an effective dimension, which is small in real-world graphs, and propose two algorithms for solving our problem that scale linearly and sublinearly in this dimension. Our experiments on real-world content recommendation problem show that a good estimator of user preferences for thousands of items can be learned from just tens of nodes evaluations.

1. Introduction

A smooth graph function is a function on a graph that returns similar values on neighboring nodes. This concept arises frequently in manifold and semi-supervised learning (Zhu, 2008), and reflects the fact that the outcomes on the neighboring nodes tend to be similar. It is well-known (Belkin et al., 2006; 2004) that a smooth graph function can be expressed as a linear combination of the eigenvectors of the graph Laplacian with smallest eigenvalues. Therefore, the problem of learning such function can be cast as a regression problem on these eigenvectors. This is the first work that brings this concept to bandits. In particular, we study a bandit problem where the arms are the nodes of a graph and the expected payoff of pulling an arm is a smooth function on this graph.

We are motivated by a range of practical problems that involve graphs. One application is targeted advertisement in social networks. Here, the graph is a social network and our goal is to discover a part of the network that is interested in a given product. Interests of people in a social network tend to change smoothly (McPherson et al., 2001), because friends tend to have similar preferences. Therefore, we take advantage of this structure and formulate this problem as learning a smooth preference function on a graph.

Another application of our work are recommender systems (Jannach et al., 2010). In content-based recommendation (Chau et al., 2011), the user is recommended items that are similar to the items that the user rated highly in the past. The assumption is that users prefer similar items similarly. The similarity of the items can be measured for instance by a nearest neighbor graph (Billsus et al., 2000), where each item is a node and its neighbors are the most similar items.

---

In this paper, we consider the following learning setting. The graph is known in advance and its edges represent the similarity of the nodes. At time $t$, we choose a node and then observe its payoff. In targeted advertisement, this may correspond to showing an ad and then observing whether the person clicked on the ad. In content-based recommendation, this may correspond to recommending an item and then observing the assigned rating. Based on the payoff, we update our model of the world and then the game proceeds into time $t+1$. Since the number of nodes $N$ can be huge, we are interested in the regime when $t < N$.

If the smooth graph function can be expressed as a linear combination of $k$ eigenvectors of the graph Laplacian, and $k$ is small and known, our learning problem can be solved using ordinary linear bandits (Auer, 2002; Li et al., 2010). In practice, $k$ is problem specific and unknown. Moreover, the number of features $k$ may approach the number of nodes $N$. Therefore, proper regularization is necessary, so that the regret of the learning algorithm does not scale with $N$. We are interested in the setting where the regret is independent of $N$ and therefore this problem is non-trivial.

2. Setting

Let $G$ be the given graph with the set of nodes $V$ and denote $|V| = N$ the number of nodes. Let $W$ be the $N \times N$ matrix of similarities $w_{ij}$ (edge weights) and $D$ is the $N \times N$ diagonal matrix with the entries $d_{ii} = \sum_j w_{ij}$. The graph Laplacian of $G$ is defined as $L = D - W$. Let $\{\lambda_k^G, q_k\}_{k=1}^N$ be the eigenvalues and eigenvectors of $L$ ordered such that $0 = \lambda_1^G \leq \lambda_2^G \leq \cdots \leq \lambda_N^G$. Equivalently, let $L = Q \Lambda Q^T$ be the eigendecomposition of $L$, where $Q$ is an $N \times N$ orthogonal matrix with eigenvectors in columns.

In our setting we assume that the reward function is a linear combination of the eigenvectors. For any set of weights $\alpha$ let $f_\alpha : V \rightarrow \mathbb{R}$ be the function defined on nodes, linear in the basis of the eigenvectors of $L$:

$$f_\alpha(v) = \sum_{k=1}^N \alpha_k (q_k)_v = \langle x_v, \alpha \rangle,$$

where $x_v$ is the $v$-th row of $Q$, i.e., $(x_v)_i = (q_i)_v$. If the weight coefficients of the true $\alpha^*$ are such that the large coefficients correspond to the eigenvectors with the small eigenvalues and vice versa, then $f_\alpha^*$ would be a smooth function on $G$ (Belkin et al., 2006). Figure 1 displays first few eigenvectors of the Laplacian constructed from data we use in our experiments. In the extreme case, the true $\alpha^*$ may be of the form $[\alpha_1^*, \alpha_2^*, \ldots, \alpha_k^*, 0, 0, \ldots, 0]_N$ for some $k \ll N$. Had we known $k$ in such case, the known linear bandits algorithm would work with the performance scaling with $k$ instead of $D = N$. Unfortunately, first, we do not know $k$ and second, we do not want to assume such an extreme case (i.e., $\alpha_i^* = 0$ for $i > k$). Therefore, we opt for the more plausible assumption that the coefficients with the high indexes are small. Consequently, we deliver algorithms with the performance that scale with the smoothness with respect to the graph.

The learning setting is the following. In each time step $t \leq T$, the recommender $\pi$ chooses a node $\pi(t)$ and obtains a noisy reward such that:

$$r_t = \langle x_{\pi(t)}, \alpha^\ast \rangle + \varepsilon_t,$$

where the noise $\varepsilon_t$ is assumed to be $R$-sub-Gaussian for any $t$. In our setting, we have $x_v \in \mathbb{R}^D$ and $\|x_v\|_2 \leq 1$ for all $x_v$. The goal of the recommender is to minimize the cumulative regret with respect to the strategy that always picks the best node w.r.t. $\alpha^*$. Let $\pi(t)$ be the node picked (referred to as pulling an arm) by an algorithm $\pi$ at time $t$. The cumulative (pseudo) regret of $\pi$ is defined as:

$$R_T = T \max_v f_{\alpha^*}(v) - \sum_{t=1}^T f_{\alpha^*}(\pi(t))$$

We call this bandit setting spectral since it is built on the spectral properties of a graph. Compared to the linear and multi-arm bandits, the number of arms $K$ is equal to the number of nodes $N$ and to the dimension of the basis $D$ (eigenvectors are of dimension $N$). However, a regret that scales with $N$ or $D$ that can be obtained using those settings is not acceptable because the number of nodes can be large. While we are mostly interested in the setting with $K = N$, our algorithms and analyses can be applied for any finite $K$.

3. Related Work

Most related setting to our work is that of the linear and contextual linear bandits. Auer (2002) proposed a SupLinRel algorithm and showed that it obtains $\sqrt{DT}$ regret that matches the lower bound by Dani et al. (2008). First practical and empirically successful algorithm was LinUCB (Li et al., 2010). Later, Chu et al. (2011) analyzed SupLinUCB, which is a LinUCB equivalent of SupLinRel, to show that it also obtains $\sqrt{DT}$ regret. Abbasi-Yadkori et al. (2011) proposed the OFUL algorithm for linear bandits which obtains $D \sqrt{T}$ regret. Using their analysis, it is possible to show that LinUCB obtains $D \sqrt{T}$ regret as well (Remark 2). Whether LinUCB matches the $\sqrt{DT}$ lower bound for this setting is still an open problem.

Abernethy et al. (2008) and Bubeck et al. (2012) studied a more difficult adversarial setting of linear bandits where the reward function is time-dependent. It is an open problem if this approaches would work in our setting and have an upper bound on the regret that scales better than with $D$.

Kleinberg et al. (2008); Slivkins (2009), and Bubeck et al. (2011) use similarity information between the context of arms, assuming a Lipschitz or more general properties.
While such settings are indeed more general, the regret bounds scale worse with the relevant dimensions. Srinivas et al. (2010) and Valko et al. (2013) also perform maximization over the smooth functions that are either sampled from a Gaussian process prior or have a small RKHS norm. Their setting is also more general than ours since it already generalizes linear bandits. However their regret bound in the linear case also scales with $D$. Moreover, the regret of these algorithms also depends on a quantity for which data-independent bounds exists only for some kernels, while our effective dimension is always computable given the graph.

Another bandit graph setting called the gang of bandits was studied by Cesa-Bianchi et al. (2013) where each node is a linear bandit with its own weight vector which are assumed to be smooth on the graph. Next, Caron et al. (2012) assume that they obtain the reward not only from the selected node but also from all its neighbors. Yet another kind of the partial observability model for bandits on graphs in the adversarial setting is considered by Alon et al. (2013).

4. Spectral Bandits

In our setting, the arms are orthogonal to each other. Thinking that the reward observed for an arm does not provide any information for other arms would not be correct because of the assumption that under another basis, the unknown parameter has a low norm. This provides additional information across the arms through the estimation of the known parameter has a low norm. This provides additional information for the arms.

Thinking that the reward observed for an arm does not provide any information for other arms would not be correct because of the assumption that under another basis, the unknown parameter has a low norm. This provides additional information across the arms through the estimation of the known parameter has a low norm. This provides additional information for the arms.

In order to present our algorithms and analyses, we introduce a notion of effective dimension $d$. We keep using capital $D$ to denote the ambient dimension, which is equal to $N$ in the spectral setting. Intuitively, the effective dimension is a proxy for the number of relevant dimensions. We first provide a formal definition and then discuss its properties.

In general, we assume there exists a diagonal matrix $\Lambda$ with the entries $0 < \lambda = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$ and a set of $K$ vectors $x_1, \ldots, x_K \in \mathbb{R}^N$ such that $\|x_i\|_2 \leq 1$ for all $i$. For the spectral bandits, we have $K = N$. Moreover, since $Q$ is an orthonormal matrix, $\|x_i\|_2 = 1$. Finally, since the first eigenvalue of a graph Laplacian is always zero, $\lambda_1^2 = 0$, we use $\Lambda = \Lambda_\mathcal{G} + \lambda I$, in order to have $\lambda_1 = \lambda$.

Definition 1. Let the effective dimension $d$ be the largest $d$ such that:

$$
(d - 1) \lambda_d \leq \frac{T}{\log(1 + T/\lambda)}
$$

The effective dimension $d$ is small when the coefficients $\lambda_i$ grow rapidly above $T$. This is the case when the dimension of the space $D$ (and $K$) is much larger than $T$, such as in graphs from social networks with very large number of nodes $N$. In contrast, when the coefficients are all small then $d$ may be of the order of $T$, which would make the regret bounds useless. Figure 2 shows how $d$ behaves compared to $D$ on the generated and the real Flixster network graphs\(^1\) that we use for the experiments in Section 6.

\(^1\)We set $\Lambda$ to $\Lambda_\mathcal{G} + \lambda I$ with $\lambda = 0.01$, where $\Lambda_\mathcal{G}$ is the graph Laplacian of the respective graph.
Spectral Bandits

Algorithm 1 SpectralUCB

Input:
\( N \): the number of nodes, \( T \): the number of pulls
\( \{ \Lambda_L, Q \} \) spectral basis of \( \mathcal{L} \)
\( \lambda, \delta \): regularization and confidence parameters
\( R, C \): upper bounds on the noise and \( \| \alpha^* \|_\Lambda \)

Run:
\( \Lambda \leftarrow \Lambda_L + \lambda I \)
\( d \leftarrow \max \{ d : (d-1) \lambda_d \leq T/\log(1+1/\lambda) \} \)
\( c \leftarrow 2R \sqrt{d \log(1+1/\lambda)} + 2 \log(1/\delta) + C \)
for \( t = 1 \) to \( T \) do
  Update the basis coefficients \( \hat{\alpha} \):
  \( X_t \leftarrow [x_1, \ldots, x_{t-1}]^T \)
  \( r_t \leftarrow [r_1, \ldots, r_{t-1}]^T \)
  \( V_t \leftarrow X_t X_t + \Lambda \)
  \( \hat{\alpha}_t \leftarrow V_t^{-1} X_t r_t \)
  Choose the node \( v_t \) (\( x_{v_t} \)-th row of \( Q \)):
  \( v_t \leftarrow \arg \max_v \left( f_{\hat{\alpha}_t}(v) + c\|x_v\|_{V_t^{-1}} \right) \)
  Observe the reward \( r_t \)
end for

4.2. SpectralUCB

The first algorithm we present is SpectralUCB (Algorithm 1) which is based on LinUCB and uses the spectral penalty (1). For clarity, we set \( x_t = x_{v_t} = x_{v(t)} \). Here we consider regularized least-squares estimate \( \hat{\alpha}_t \) of the form:

\[
\hat{\alpha}_t = \arg \min_{\alpha} \left( \sum_{u=1}^{t} \| x_u, \alpha - r_u \|^2 + \| \alpha \|^2 \right)
\]

A key part of the algorithm is to define the \( c_t \) confidence widths for the predictions of the rewards. We take advantage of our analysis (Section 5.2) to define \( c_t \) based on the effective dimension \( d \) which is specifically tailored to our setting. By doing this we also avoid the computation of the determinant (see Section 5). The following theorem characterizes the performance of SpectralUCB and bounds the regret as a function of effective dimension \( d \).

Theorem 1. Let \( d \) be the effective dimension and \( \lambda \) be the minimum eigenvalue of \( \Lambda \). If \( \| \alpha^* \|_\Lambda \leq C \) and for all \( x_v, \langle x_v, \alpha^* \rangle \in [1,1], \) then the cumulative regret of SpectralUCB is with probability at least \( 1 - \delta \) bounded as:

\[
R_T \leq \left[ 8R \sqrt{d \log(1+T/\lambda)} + 2 \log(1/\delta) + 4C \right] \times dT(1+1/\lambda) \log(1+T/\lambda)
\]

Remark 1. The constant \( C \) needs to be such that \( \| \alpha^* \|_\Lambda \leq C \). If we set \( C \) too small, the true \( \alpha^* \) will lie outside of the region and far from \( \hat{\alpha}_t \), causing the algorithm to underperform. Alternatively, \( C \) can be time dependent, e.g., \( C_t = \log T \). In such case, we do not need to know an upper bound on \( \| \alpha^* \|_\Lambda \) in advance, but our regret bound would only hold after some \( t \), when \( C_t \geq \| \alpha^* \|_\Lambda \).

We provide the proof of Theorem 1 in Section 5 and examine the performance of SpectralUCB experimentally in Section 6. The \( d \sqrt{T} \) result of Theorem 1 is to be compared with the classical linear bandits, where LinUCB is the algorithm often used in practice (Li et al., 2010) achieving \( D \sqrt{T} \) cumulative regret. As mentioned above and demonstrated in Figure 2, in the \( T < N \) regime we can expect \( d \ll D = N \) and obtain an improved performance.

4.3. SpectralEliminator

It is known that the available upper bound for LinUCB or OFUL is not optimal for the linear bandit setting with finite number of arms in terms of dimension \( D \). On the other hand, the algorithms SupLinRel or SupLinUCB achieve the optimal \( \sqrt{DT} \) regret. In the following, we likewise provide an algorithm that also scales better with \( d \) and achieves \( \sqrt{dT} \) regret. The algorithm is called SpectralEliminator (Algorithm 2) and works in phases, eliminating the arms that are not promising. The phases are defined by the time indexes \( t_1 = 1 \leq t_2 \leq \ldots \) and depend on some parameter \( \beta \). The algorithm is in a spirit similar to the Improved UCB by Auer & Ortner (2010). The main idea of SpectralEliminator is to divide the time steps in to sets in order to introduce independence and allow the Azuma-Hoeffding inequality (Azuma, 1967) to be applied. In the following theorem we characterize the performance of SpectralEliminator and show that the upper bound on regret has \( \sqrt{d} \) improvement over SpectralUCB.

Algorithm 2 SpectralEliminator

Input:
\( N \): the number of nodes, \( T \): the number of pulls
\( \{ \Lambda_L, Q \} \) spectral basis of \( \mathcal{L} \)
\( \lambda \): regularization parameter
\( \beta, \{ r_j \} \): parameters of the elimination and phases
\( A_t \leftarrow \{ x_1, \ldots, x_K \} \)
for \( j = 1 \) to \( J \) do
  \( V_j \leftarrow \gamma \Lambda_L + \lambda I \)
for \( t = t_j \) to \( \min(t_{j+1} - 1, T) \) do
  Play \( x_t \in A_j \) with the largest width to observe \( r_t \):
  \( x_t \leftarrow \arg \max_{x \in A_j} \| x \|_{V_t^{-1}} \)
  \( V_{t+1} \leftarrow V_t + x_t x_t^T \)
end for

Eliminate the arms that are not promising:
\[
\hat{\alpha}_j \leftarrow V_{t+1}^{-1} [x_{t_j}, \ldots, x_t] [r_{t_j}, \ldots, r_t]^T
\]
\[
p \leftarrow \max_{x \in A_j} \left[ \hat{\alpha}_j, x \right] - \| x \|_{V_{t+1}^{-1} \beta} \geq p
\]
\( A_{j+1} \leftarrow \{ x \in A_j, (\hat{\alpha}_j, x) + \| x \|_{V_{t+1}^{-1} \beta} \geq p \} \)
end for
Theorem 2. Choose the phases starts as $t_j = 2^{j-1}$. Assume all rewards are in $[0, 1]$ and $\|x^\star\|_A \leq C$. For any $\delta > 0$, with probability at least $1 - \delta$, the cumulative regret of SPECTRAL ELIMINATOR algorithm run with parameter $\beta = 2R\sqrt{14\log(2K(1 + \log_2 T))/\delta} + C$ is bounded as:

$$R_T \leq 2 + 32 \left(2R\sqrt{\frac{14 \log(2K(1 + \log_2 T))}{\delta}} + C \right) \times \sqrt{2dT(1 + 1/\lambda) \log (1 + T/\lambda)}$$

4.4. Scalability and computational complexity

There are three main computational issues to address in order to make SPECTRAL UCB scalable: the computation of $N$ UCBs, matrix inversion, and obtaining the eigenbasis which serves as an input to the algorithm. First, to speed up the computation of $N$ UCBs in each time step, we use lazy updates technique (Desautels et al., 2012) which maintains a sorted queue of UCBs and in practice leads to substantial speed gains. Second, to speed up matrix inversion we do iterative matrix inversion (Zhang, 2005).

Finally, while the eigendecomposition of a general matrix is computationally difficult, Laplacians are symmetric diagonally dominant (SDD). This enables us to use fast SDD solvers as CMG by Koutsis et al. (2011). Furthermore, using CMG we can find good approximations to the first $L$ eigenvectors in $O(Lm \log m)$ time, where $m$ is the number of edges in the graph (e.g. $m = 10N$ in the Flixter experiment). CMG can easily work with $N$ in millions. In general, we have $L = N$ but from our experience, a smooth reward function can be often approximated by dozens of eigenvectors. In fact, $L$ can be considered as an upper bound on the number of eigenvectors we actually need. Furthermore, by choosing small $L$ we not only reduce the complexity of eigendecomposition but also the complexity of the least-square problem being solved in each iteration.

Choosing a small $L$ can significantly reduce the computation but it is important to choose $L$ large enough so that still less than $J$ eigenvectors are enough. This way, the problem that we solve is still relevant and our analysis applies. In short, the problem cannot be solved trivially by choosing first $k$ relevant eigenvectors because $k$ is unknown. Therefore, in practice we choose the largest $L$ such that our method is able to run. In Section 6.4, we demonstrate that we can obtain good results with relatively small $L$.

5. Analysis

The analysis of SPECTRAL UCB (Section 5.3) has two main ingredients. The first one is the derivation of the confidence ellipsoid for $x^\star$, which is a straightforward update of the analysis of OFUL (Abbasi-Yadkori et al., 2011) using self-normalized martingale inequality (Section 5.1). The second part is crucial to prove that the final regret bound scales only with the effective dimension $d$ and not with the ambient dimension $D$. We achieve this by considering the geometrical properties of the determinant which hold in our setting (Section 5.2). We also used this result to upper-bound the regret of SPECTRAL ELIMINATOR (Section 5.4). The proofs of the lemmas are in the appendix.

5.1. Confidence ellipsoid

The first lemma is by Abbasi-Yadkori et al. (2011) and we restate them for the convenience.

Lemma 1. Let $V_t = \Lambda + \sum_{s=1}^{t-1} x_s x_s^\top$ and define $\xi_t = \sum_{s=1}^{t-1} e_s x_s$. With probability at least $1 - \delta$, $\forall t \geq 1$:

$$\|\xi_t\|_{V_t^{-1}}^2 \leq 2R^2 \log \left(\frac{|V_t|^{1/2}}{\delta |\Lambda|^{1/2}}\right)$$

The next lemma is a generalization of Theorem 2 by Abbasi-Yadkori et al. (2011) to the regularization with $\Lambda$. The result of this lemma is also used for the confidence coefficient $c_t$ in Algorithm 1, which we upperbound in Section 5.2 to avoid the computation of determinants.

Lemma 2. Let $V_t = \Lambda + \sum_{s=1}^{t-1} x_s x_s^\top$ and $\|x^\star\|_A \leq C$. With probability at least $1 - \delta$, for any $x$ and $t \geq 1$:

$$|x^\top (\hat{x}_t - x^\star)| \leq \|x\|_{V_t^{-1}} \left(2R\sqrt{\frac{2\log \left(\frac{|V_t|^{1/2}}{\delta |\Lambda|^{1/2}}\right)}{C} + C} \right)$$

To control the norm $\|x\|_{V_t^{-1}}$ we have the following lemma.

Lemma 3. For all $t$,

$$\|x_t\|_{V_t^{-1}} \leq \left(2 + \frac{2}{\lambda}\right) \log \left(1 + \|x_t\|_{V_t^{-1}}^2\right).$$

5.2. Effective dimension

In Section 5.1 we show that several quantities scale with $\log(|V_t|/|\Lambda|)$, which can be of the order of $D$. Therefore, in this part we present the key ingredient of our analysis based on the geometrical properties of determinants (Lemmas 4 and 5) to upperbound $\log(|V_t|/|\Lambda|)$ by a term that scales with $d$ (Lemma 6). Not only this will allow us to show that the regret scales with $d$, but it also helps us to avoid the computation of the determinants in Algorithm 1.

Lemma 4. Let $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_N)$ be any diagonal matrix with strictly positive entries and for any vectors $\{x_s\}_{1 \leq s < t}$ such that $\|x_s\|_2 \leq 1$ for all $1 \leq s < t$, we have that the determinant $|V_t|$ of $V_t = \Lambda + \sum_{s=1}^{t-1} x_s x_s^\top$ is maximized when all $x_s$ are aligned with the axes.

Lemma 5. Let $V_t = \Lambda + \sum_{s=1}^{t-1} x_s x_s^\top$. Then:

$$\log \left(\frac{|V_t|}{|\Lambda|}\right) \leq \max_{i=1}^N \sum_{i=1}^N \log \left(1 + \frac{t_i}{x_i}\right),$$

where the maximum is taken over all possible positive real numbers $\{t_1, \ldots, t_N\}$, such that $\sum_{i=1}^N t_i = t - 1$. 

Spectral Bandits
Lemma 6. Let $d$ be the effective dimension and $T$ be the time horizon of the algorithm. Then:

$$\log \frac{|V_{T+1}|}{|A|} \leq 2d \log \left(1 + \frac{T}{\lambda}\right)$$

5.3. Cumulative regret of SPECTRALUCB

Proof of Theorem 1. Let $x_s = \arg \max_{x_s} x_s^T \alpha^*$ and let $\tilde{r}_t$ denote the instantaneous regret at time $t$. With probability at least $1 - \delta$, for all $t$:

$$\tilde{r}_t = x^T_s \alpha^* - x^T_s \alpha^*$$

$$\leq x^T_s \hat{\alpha}_t + c \|x_s\|_{\tilde{V}^{-1}_t} - x^T_s \alpha^* \quad (2)$$

$$\leq x^T_s \hat{\alpha}_t + c \|x_s\|_{\tilde{V}^{-1}_t} - x^T_s \alpha + c \|x_s\|_{\tilde{V}^{-1}_t} \quad (3)$$

$$= 2c \|x_s\|_{\tilde{V}^{-1}_t}.$$

The inequality (2) is by the algorithm design and reflects the optimistic principle of SPECTRALUCB. Specifically, $x^T_s \hat{\alpha}_t + c \|x_s\|_{\tilde{V}^{-1}_t} \leq x^T_s \alpha + c \|x_s\|_{\tilde{V}^{-1}_t}$, from which:

$$x^T_s \alpha^* \leq x^T_s \hat{\alpha}_t + c \|x_s\|_{\tilde{V}^{-1}_t} \leq x^T_s \alpha + c \|x_s\|_{\tilde{V}^{-1}_t}.$$

In (3) we applied Lemma 2: $x^T_s \hat{\alpha}_t \leq x^T_s \alpha + c \|x_s\|_{\tilde{V}^{-1}_t}$. Finally, by Lemmas 3 and 6:

$$R_T = \sum_{t=1}^{T} \tilde{r}_t \leq 2c \sum_{t=1}^{T} \|x_s\|_{\tilde{V}^{-1}_t} \leq 2c \sqrt{T \sum_{t=1}^{T} \|x_s\|^2_{\tilde{V}^{-1}_t}}$$

$$\leq 2c \sqrt{T \left(2 + \frac{2}{\lambda}\right) \sum_{t=1}^{T} \log \left(1 + \|x_s\|^2_{\tilde{V}^{-1}_t}\right)}$$

$$= 2c \sqrt{T \left(2 + \frac{2}{\lambda}\right) \log \frac{|V_{T+1}|}{|A|}}$$

$$\leq 4c \sqrt{d} T \left(1 + \frac{1}{\lambda}\right) \log \left(1 + \frac{T}{\lambda}\right)$$

By plugging $c$, we get that with probability at least $1 - \delta$, the theorem holds.

Remark 2. Notice that if we set $A = I$ in Algorithm 1, we recover LinUCB. Since $\log(|V_{T+1}|/|A|)$ can be upperbounded by $D \log T$ (Abbasi-Yadkori et al., 2011), we obtain $O(D \sqrt{T})$ upper bound of regret of LinUCB as a corollary of Theorem 1.

5.4. Cumulative regret of SPECTRAELIMINATOR

The probability space induced by the rewards $r_1, r_2, \ldots$ can be decomposed as a product of independent probability spaces induces by rewards in each phase $[t_j, t_{j+1} - 1]$. Denote by $F_j$ the $\sigma$-algebra generated by the rewards $r_1, \ldots, r_{t_{j+1} - 1}$, i.e., received before and during the phase $j$. We have the following two lemmas for any phase $j$. Let $V_j = A + \sum_{t_j}^{t_{j+1}-1} x_s x_s^T$ and $\hat{\alpha}_j$ for $\alpha_j$.

Lemma 7. For any fixed $x \in \mathbb{R}^N$ and any $\delta > 0$, we have that if $\beta(\alpha^*, \delta) = 2R \sqrt{14 \log(2/\delta) + \|\alpha^*\|^2_A}$, then at time $t_j$ (beginning of phase $j$):

$$\mathbb{P} \left[ \left| x^T (\hat{\alpha}_j - \alpha^*) \right| \leq \|x\|_{V_j^{1/2}} \beta(\alpha^*, \delta) \right] \geq 1 - \delta$$

Lemma 8. For all $x \in A_j$, $j > 1$, we have:

$$\|x\|^2_{V_j^{-1}} \leq \frac{1}{t_j - t_{j-1}} \sum_{s=t_{j-1}}^{t_{j-1}} \|x_s\|^2_{V_s^{-1}}$$

Now we are ready to upperbound the cumulative regret of SPECTRAELIMINATOR.

Proof of Theorem 2. Let $J = \lfloor \log_2 T \rfloor + 1$ and $t_j = 2^j - 1$. We have:

$$R_T = \sum_{t=1}^{T} \langle x_s - x_t, \alpha^* \rangle \leq 2 + \sum_{j=2}^{J} \sum_{t_{j-1}}^{t_{j-1}} \langle x_s - x_t, \alpha^* \rangle$$

$$\leq 2 + \sum_{j=2}^{J} \sum_{t_{j-1}}^{t_{j-1}} \left[ \|x_s - x_t, \hat{\alpha}_j \| \right]$$

$$+ \left( \|x_s\|_{V_j^{-1}} + \|x_t\|_{V_j^{-1}} \right) \beta(\alpha^*, \delta'),$$

in an event $\Omega$ of probability $1 - \delta$, where we used Lemma 7 in the last inequality for $\delta' = \delta/(KJ)$. By definition of the action subset $A_j$ at phase $j > 1$, under $\Omega$, we have:

$$\langle x_s - x_t, \hat{\alpha}_j \rangle \leq \left( \|x_s\|_{V_j^{-1}} + \|x_t\|_{V_j^{-1}} \right) \beta(\alpha^*, \delta'),$$

since $x_s \in A_j$ for all $j \leq J$. By previous two lemmas:

$$R_T \leq 2 + 4\beta(\alpha^*, \delta') \sum_{j=2}^{J} \left( t_{j+1} - t_j \right)^2 \sum_{s=t_{j-1}}^{t_{j-1}} \|x_s\|^2_{V_s^{-1}}$$

$$\times \left( 2 \left( 1 + \frac{2}{\lambda}\right)^{\log \left(1 + \|x_s\|^2_{V_s^{-1}}\right)} \right)$$

$$\leq 2 + 4\beta(\alpha^*, \delta') \sum_{j=2}^{J} \left( \frac{t_{j+1} - t_j}{t_j - t_{j-1}} \right)^2 \sum_{s=t_{j-1}}^{t_{j-1}} \|x_s\|^2_{V_s^{-1}}$$

$$\leq 2 + 4\beta(\alpha^*, \delta') \left( \frac{2^j \left( 2^j - 2 \right)}{2^{j-2}} \right)^{\log \left|V_j\right|}$$

$$\leq 2 + 4\beta(\alpha^*, \delta') \left( \frac{2}{\lambda} \right)^{2d} \log \left(1 + \frac{T}{\lambda}\right) \sum_{j=2}^{J} 2^j$$

Finally, using $\beta(\alpha^*, \delta') < \beta(\alpha^*, \delta'/K(1 + |\log_2 T|))$ and $\delta' = \delta/(KJ)$, we obtain the result of Theorem 2. \qed
Remark 3. If we set $\Lambda = \mathbf{I}$ in Algorithm 2 as in Remark 2, we get a new algorithm, LINEARELIMINATOR, which is a competitor to SupLinRel (Auer, 2002) and as a corollary to Theorem 2 also enjoys $O(\sqrt{DT})$ upper bound on the cumulative regret. On the other hand, compared to SupLinRel, LINEARELIMINATOR and its analysis are significantly much simpler and elegant.

6. Experiments

We evaluated our algorithms and compared them to LinUCB. In all experiments we set $\delta$ to 0.001 and $R$ to 0.01. For SpectralUCB and SpectralEliminator we set $\Lambda$ to $\Lambda_\ell + \lambda \mathbf{I}$ with $\lambda = 0.01$. For LinUCB we regularized with $\lambda \mathbf{I}$ with $\lambda = 1$. Our results are robust to small perturbations of all learning parameters. We also performed experiments with SupLinRel, SupLinUCB, SupSpectralUCB, but due to the known reasons (Chu et al., 2011) these algorithms are not efficient\footnote{an equivalent of SupLinUCB with spectral regularization} and they were always outperformed by SpectralUCB and LinUCB.

6.1. Random graph models

To simulate realistic graph structures, we generated graphs of $N$ nodes using three models that are commonly used in the social networks modeling. First, we considered the widely known Erdős-Rényi (ER) model. We sampled the edges in the ER model independently with probability $3\%$. Second, we considered the Barabási-Albert (BA) model (1999), with the degree parameter 3. BA models are commonly used for modeling real networks due to their preferential attachment property. Finally, we considered graphs where the edge structure forms a regular lattice.

For all the graph models we assigned uniformly random weights to their edges. Then, we randomly generated $k$-sparse vector $\alpha^*$ of $N$ weights, $k \ll N$ and defined the true graph function as $f = \mathbf{Q}\alpha^*$, where $\mathbf{Q}$ is the matrix of eigenvectors from the eigendecomposition of the graph Laplacian. We ran the algorithms in the desired $T < N$ regime, with $N = 500$ ($N = 5^4$ for the lattice), $T = 250$, and $k = 5$. Figure 3 shows that the regret of LinUCB for all three models has within first $T$ steps still a linear trend unlike SpectralUCB that performs much better.

Unfortunately, even though the regret bound Spectral Eliminator is asymptotically better, it was outperformed by SpectralUCB. This is similar to the linear case when LinUCB outperforms\footnote{For example, these algorithms do not use all the rewards obtained for the estimation of $\hat{\alpha}$.} SupLinUCB in practice (Chu et al., 2011) while it is open problem whether LinUCB can be shown to have a better regret.

6.2. MovieLens experiments

In this experiment we took user preferences and the similarity graph over movies from the MovieLens dataset (Lam & Herlocker, 2012), a dataset of 6k users who rated one million movies. We divide the dataset into two equally-sized parts. The first dataset is used to build our model of users, the rating that user $i$ assigns to movie $j$. We factor the user-item matrix using low-rank matrix factorization (Keshavan et al., 2009) as $\mathbf{M} \approx \mathbf{U}\mathbf{V}^T$, a standard approach to collaborative filtering. The rating that the user $i$ assigns to movie $j$ is estimated as $\hat{r}_{i,j} = \langle \mathbf{u}_i, \mathbf{v}_j \rangle$, where $\mathbf{u}_i$ is the $i$-th row of $\mathbf{U}$ and $\mathbf{v}_j$ is the $j$-th row of $\mathbf{V}$. The rating $\hat{r}_{i,j}$ is the payoff of pulling arm $j$ when recommending to user $i$. The second dataset is used to build our similarity graph over movies. We factor the dataset in the same way as the first dataset. The graph contains an edge between movies $i$ and $i'$ if the movie $i'$ is among 10 nearest neighbors of the movie $i$ in the latent space of items $\mathbf{V}$. The weight on all edges is one. Notice that if two items are close in the item space, then their expected rating is expected to be similar. However, the opposite is not true. If two items have a similar expected rating, they do not have to be close in the item space. In other words, we take advantage of ratings but do not hardwire the two similarly rated items to be similar.

In Figure 4, we sampled 50 users and evaluated the regret of both algorithms for $T = 100$. Here SpectralUCB suffers only about one fourth of regret over LinUCB, specifically 43.4611 vs. 133.0996 on average.
6.3. Flixter Experiments

We also performed experiments on users preferences from the movie recommendation website Flixter. The social network of the users was crawled by Jamali & Ester (2010) and then clustered by Graclus (2013) to obtain a strongly connected subgraph. We extracted a subset of users and movies, where each movie has at least 30 ratings and each user rated at least 30 movies. This resulted in a dataset of 4546 movies and 5202 users. As with MovieLens dataset we completed the missing ratings by a low-rank matrix factorization and used it construct a 10-NN similarity graph.

Again in Figure 4, we sampled 50 users and evaluated the regret of both algorithms for $T = 100$. On average, \textsc{SpectralUCB} suffers only about one third of regret over LinUCB, specifically 37.6499 vs. 99.8730 on average.

6.4. Reduced basis

As discussed in Section 4.4, one can decrease the computational complexity and thus increase the scalability by only extracting first $L \ll N$ eigenvectors of the graph Laplacian. First, the computational complexity of such operation is $O(Lm \log m)$, where $m$ is the number of edges. Second, the least-squares problem that we have to do in each time step of the algorithm is only $L$ dimensional.

In Figure 5 we plot the cumulative regret and the total computational time in seconds (log scale) for a single user from the MovieLens dataset. We varied $L$ as 20, 200, and 2000 which corresponds to about 1%, 10% and 100% of basis functions ($N = 2019$). The total computational time also includes the computational savings from lazy updates and iterative matrix inversion. We see that with 10% of the eigenvectors we can achieve similar performance as for the full set for the fraction of the computational time.

7. Conclusion

We presented spectral bandit setting inspired mostly by the applications in the recommender systems and targeted advertisement in social networks. In this setting, we are asked to repeatedly maximize an unknown graph function, assumed to be smooth on a given similarity graph. Traditional linear bandit algorithm can be applied but their regret scales with the ambient dimension $D$, either linearly or as a square root, which can be very large.

Therefore, we introduced two algorithms, \textsc{SpectralUCB} and \textsc{SpectralEliminator}, for which the regret only scales with effective dimension $d$ which is typically much smaller than $D$ for real-world graphs. We demonstrated that \textsc{SpectralUCB} delivers desired benefit for the graphs generated by Barabási–Albert, Erdős-Rényi, and regular lattice models; and for the movie rating data from the MovieLens and Flixter social networks. In future, we plan to extend this work to a sparse setting when the smooth function is assumed to be a linear combination of only finite number of eigenvectors.

8. Acknowledgements

We would like to thank Yiannis Koutis for his great help with the efficient computation of eigenvectors. We thank Andreas Krause for suggesting the lazy updates of UCBs. We would also like to thank Giovanni Zappella, Claudio Gentile, and especially Alessandro Lazaric for helpful discussions. The research presented in this paper was supported by French Ministry of Higher Education and Research, by European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement n°270327 (project CompLACS), and by Intel Corporation.
Spectral Bandits

References


Supplementary Material

Confidence ellipsoid

Lemma 2. Let $V_t = \Lambda + \sum_{s=1}^{t-1} x_s x_s^T$ and $\|\alpha^*\|_\Lambda \leq C$.
With probability at least $1 - \delta$, for any $x$ and $t \geq 1$:

$$|x^T(\hat{\alpha}_t - \alpha^*)| \leq \|x\|_{V^{-1}_t} \left(R\sqrt{2\log \left(\frac{|V_t|^{1/2}}{\delta\Lambda^{1/2}}\right)} + C\right)$$

Proof of Lemma 2. We have:

$$|x^T(\hat{\alpha}_t - \alpha^*)| = |x^T(-V_t^{-1}\Lambda\alpha^* + V_t^{-1}\xi_t)|$$

$$\leq |x^T V_t^{-1}\Lambda\alpha^*| + |x^T V_t^{-1}\xi_t|$$

$$\leq \langle x^T, \Lambda\alpha^* \rangle_{V_t^{-1}} + \langle x, \xi_t \rangle_{V_t^{-1}}$$

$$\leq \|x\|_{V_t^{-1}} \left(\|\xi_t\|_{V_t^{-1}} + \|\Lambda\alpha^*\|_{V_t^{-1}}\right),$$

where we used Cauchy-Schwarz inequality in the last step. Now we bound $\|\xi_t\|_{V_t^{-1}}$ by Lemma 1 and notice that:

$$\|\alpha^*\|_{V_t^{-1}} = \sqrt{(\alpha^*)^T V_t^{-1} \Lambda \alpha^*}$$

$$\leq \sqrt{(\alpha^*)^T \Lambda \alpha^*} = \|\alpha^*\|_\Lambda \leq C$$

Lemma 3. For all $t$,

$$\|x_t\|_{V_t^{-1}} \leq \left(2 + \frac{2}{\lambda}\right) \log \left(1 + \|x_t\|_{V_t^{-1}}^2\right).$$

Proof. Note that $\|x_t\|_{V_t^{-1}} \leq (1/\sqrt{\lambda}) \|x_t\| \leq (1/\sqrt{\lambda})$ and for all $0 \leq x \leq 1$ we have

$$x \leq 2\log(1 + x).$$

(4)

Now we consider two cases depending on $\lambda$. If $\lambda \geq 1$, we know that $0 \leq \|x_t\|_{V_t^{-1}} \leq 1$ and therefore by (4),

$$\|x_t\|_{V_t^{-1}}^2 \leq 2 \log \left(1 + \|x_t\|_{V_t^{-1}}^2\right).$$

Similarly, if $\lambda < 1$, then $0 \leq \lambda \|x_t\|_{V_t^{-1}} \leq 1$ and we get

$$\|x_t\|_{V_t^{-1}}^2 \leq \frac{2}{\lambda} \log \left(1 + \lambda \|x_t\|_{V_t^{-1}}^2\right)$$

$$\leq \frac{2}{\lambda} \log \left(1 + \|x_t\|_{V_t^{-1}}^2\right).$$

Combining the two, we get that for all $\lambda \geq 0$,

$$\|x_t\|_{V_t^{-1}}^2 \leq \max \left(2, \frac{2}{\lambda}\right) \log \left(1 + \|x_t\|_{V_t^{-1}}^2\right)$$

$$\leq \left(2 + \frac{2}{\lambda}\right) \log \left(1 + \|x_t\|_{V_t^{-1}}^2\right).$$

Effective dimension

Lemma 11. For any real positive-definite matrix $A$ with only simple eigenvalue multiplicities and any vector $x$ such that $\|x\|_2 \leq 1$ we have that the determinant $|A + xx^T|$ is

maximized by a vector $x$ which is aligned with an eigenvector of $A$.

Proof of Lemma 11. Using Sylvester’s determinant theorem, we have:

$$|A + xx^T| = |A||I + A^{-1}xx^T| = |A|(1 + x^T A^{-1}x)$$

From the spectral theorem, there exists an orthonormal matrix $U$, the columns of which are the eigenvectors of $A$; such that $A = UDU^T$ with $D$ being a diagonal matrix with the positive eigenvalues of $A$ on the diagonal. Thus:

$$\max_{\|x\|_2 \leq 1} x^T A^{-1}x = \max_{\|x\|_2 \leq 1} x^T U D^{-1} U^T x$$

$$= \max_{\|y\|_2 \leq 1} y^T D^{-1} y,$$

since $U$ is a bijection from $\{x, \|x\|_2 \leq 1\}$ to itself.

Since there are no multiplicities, it is easy to see that the quadratic mapping $y \mapsto y^T D^{-1} y$ is maximized (under the constraint $\|y\|_2 \leq 1$) by a canonical vector $e_I$ corresponding to the lowest diagonal entry $I$ of $D$. Thus the maximum of $x \mapsto x^T A^{-1} x$ is reached for $U e_I$, which is the eigenvector of $A$ corresponding to its lowest eigenvalue.

Lemma 4. Let $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_N)$ be any diagonal matrix with strictly positive entries. Then for any vectors $(x_s)_{1 \leq s \leq t}$, such that $\|x_s\|_2 \leq 1$ for all $1 \leq s < t$, we have that the determinant $|V_t|$ of $V_t = \Lambda + \sum_{s=1}^{t-1} x_s x_s^T$ is maximized when all $x_s$ are aligned with the axes.

Proof of Lemma 4. Let us write $d(x_1, \ldots, x_{t-1}) = |V_t|$ the determinant of $V_t$. We want to characterize:

$$\max_{x_1, \ldots, x_{t-1} : \|x_i\|_2 \leq 1, 1 \leq s < t} d(x_1, \ldots, x_{t-1})$$

For any $1 \leq s < t$, let us define:

$$V_{-i} = \Lambda + \sum_{s=1 \atop s \neq i}^{t-1} x_s x_s^T$$

We have that $V_t = V_{-i} + x_i x_i^T$. Consider the case with only simple eigenvalue multiplicities. In this case, Lemma 11 implies that $x_s \mapsto d(x_1, \ldots, x_i, \ldots, x_{s-1})$ is maximized when $x_i$ is aligned with an eigenvector of $V_{-i}$. Thus all $x_s$, for $1 \leq s < t$, are aligned with an eigenvector of $V_{-i}$ and therefore also with an eigenvector of $V_t$. Consequently, the eigenvectors of $\sum_{s=1}^{t-1} x_s x_s^T$ are also aligned.
with $V_t$. Since $\Lambda = V_t - \sum_{s=1}^{t-1} x_s x_s^T$ and $\Lambda$ is diagonal, we conclude that $V_t$ is diagonal and all $x_s$ are aligned with the canonical axes.

Now in the case of eigenvalue multiplicities, the maximum of $|V_t|$ may be reached by several sets of vectors $\{(x_s^n)_{1 \leq s \leq t}\}$ but for some $m^*$, the set $(x_s^{m^*})_{1 \leq s \leq t}$ will be aligned with the axes. In order to see that, consider a perturbed matrix $V^{-1}$ by a random perturbation of amplitude at most $\varepsilon$, i.e. such that $V^{\varepsilon}_{-i} \rightarrow V_{-i}$ when $\varepsilon \rightarrow 0$. Since the perturbation is random, then the probability that $\Lambda^*$, as well as all other $V_{-i}$ possess an eigenvalue of multiplicity bigger than 1 is zero. Since the mapping $\varepsilon \mapsto V^{\varepsilon}_{-i}$ is continuous, we deduce that any adherent point $\bar{x}_i$ of the sequence $(x_i^*)_{\varepsilon}$ (there exists at least one since the sequence is bounded in $\ell_2$-norm) is aligned with the limit $V_{-i}$ and we can apply the same reasoning.

**Lemma 5.** For any $T$, let $V_{T+1} = \sum_{t=1}^{T} x_t x_t^T + \Lambda$. Then:

$$\log \frac{|V_{T+1}|}{|\Lambda|} \leq \max\left\{ \log \left(1 + \frac{t_i}{\lambda_i}\right) \right\},$$

where the maximum is taken over all possible positive real numbers $\{t_1, \ldots, t_N\}$, such that $\sum_{i=1}^{N} t_i = T$.

**Proof of Lemma 5.** We want to bound the determinant $|V_{T+1}|$ under the coordinate constraints $\|x_t\|_2 \leq 1$. Let:

$$M(x_1, \ldots, x_T) = |\Lambda + \sum_{t=1}^{T} x_t x_t^T|$$

From Lemma 4 we deduce that the maximum of $M$ is reached when all $x_i$ are aligned with the axes:

$$M = \max_{x_1, \ldots, x_T: x_t \in \{e_1, \ldots, e_N\} } \left| \Lambda + \sum_{t=1}^{T} x_t x_t^T \right|$$

$$= \max_{t_1, \ldots, t_N \text{ positive integers}} \sum_{i=1}^{N} t_i $$

$$\leq \max_{t_1, \ldots, t_N \text{ positive reals}} \prod_{i=1}^{N} \left( \lambda_i + t_i \right),$$

from which we obtain the result.

**Lemma 6.** Let $d$ be the effective dimension and $T$ be the time horizon of the algorithm. Then:

$$\log \frac{|V_{T+1}|}{|\Lambda|} \leq 2d \log \left(1 + \frac{T}{\lambda}\right)$$

**Proof of Lemma 6.** Using Lemma 5 and Definition 1:

$$\log \frac{|V_{T+1}|}{|\Lambda|} \leq \sum_{t=1}^{d} \log \left(1 + \frac{T}{\lambda}\right) + \sum_{i=d+1}^{N} \log \left(1 + \frac{T}{\lambda_i}\right)$$

$$\leq d \log \left(1 + \frac{T}{\lambda}\right) + \sum_{i=d+1}^{N} \frac{T}{\lambda_i} \leq 2d \log \left(1 + \frac{T}{\lambda}\right)$$

**SpectralEliminator**

**Lemma 7.** For any fixed $x \in \mathbb{R}^N$ and any $\delta > 0$, we have that if $\beta(\alpha^*, \delta) = 2R \sqrt{\log(2/\delta)} + \|\alpha^*\|_A$, then at time $t_j$ (beginning of phase $j$):

$$\mathbb{P}\left( |x^T(\bar{\alpha}_j - \alpha^*)| \leq \|x\|_{V_j^{-1}} \beta(\alpha^*, \delta) \right) \geq 1 - \delta$$

**Proof of Lemma 7.** Defining $\xi_j = \sum_{s=t_j-1}^{t_j-1} x_s \varepsilon_s$, we have:

$$|x^T(\bar{\alpha}_j - \alpha^*)| = |x^T(-V_j^{-1} \Lambda \alpha^* + V_j^{-1} \xi_j)|$$

$$\leq |x^T \Lambda \alpha^*| + |x^T V_j^{-1} \xi_j| \quad (5)$$

The first term in the right hand side of (5) is bounded as:

$$|x^T \Lambda \alpha^*| \leq \|x^T \Lambda \alpha^*\| = \|\alpha^*\|_A \sqrt{x^T \Lambda \alpha^*} |x|$$

$$\leq \|\alpha^*\|_A \sqrt{x^T \Lambda \alpha^*} \leq \|\alpha^*\|_A \|x\|_{V_j^{-1}}$$

Now consider the second term in the r.h.s. of (5). We have:

$$\left| x^T V_j^{-1} \xi_j \right| = \left| \sum_{s=t_j-1}^{t_j-1} (x^T V_j^{-1} x_s) \varepsilon_s \right|$$

Let us notice that the points $(x_s)$ selected by the algorithm during phase $j - 1$ only depend on their width $\|x\|_{V_j^{-1}}$, which does not depend on the rewards received during the phase $j - 1$. Thus, given $F_{j-2}$, the sequence $(x_s)_{t_j-1 \leq s \leq t_j}$ is deterministic. Consequently, one may use a variant of Azuma’s inequality (Shamir, 2011):

$$\mathbb{P}\left( \left| x^T V_j^{-1} \xi_j \right|^2 \leq 2R^2 \log(2/\delta) \times \right.$$

$$\times x^T V_j^{-1} \left( \sum_{s=t_j-1}^{t_j-1} x_s x_s^T \right) V_j^{-1} x \left| F_{j-2} \right) \geq 1 - \delta,$$
from which we deduce:
\[ \Pr \left( \left| \mathbf{x}^\top \mathbf{\nabla}_j^{-1} \mathbf{\xi}_j \right|^2 \leq 56R^2 \mathbf{x}^\top \mathbf{\nabla}_j^{-1} \mathbf{x} \log(2/\delta) \right| \mathcal{F}_{j-2} \geq 1 - \delta, \]

since \( \sum_{s=t_{j-1}}^{t_j-1} \mathbf{x}_s^\top \mathbf{x}_s \prec \mathbf{\nabla}_j \). Thus:
\[ \Pr \left( \left| \mathbf{x}^\top \mathbf{\nabla}_j^{-1} \mathbf{\xi}_j \right| \leq 2R \left\| \mathbf{x} \right\| V_j^{-1} \sqrt{14 \log(2/\delta)} \right) \geq 1 - \delta \]

Lemma 8. For all \( \mathbf{x} \in A_j, j > 1 \), we have:
\[ \left\| \mathbf{x} \right\| V_j^{-1} \leq \frac{1}{t_j - t_{j-1}} \sum_{s=t_{j-1}}^{t_j-1} \left\| \mathbf{x}_s \right\| V_s^{-1} \]

Proof of Lemma 8. We have:
\[ (t_j - t_{j-1}) \left\| \mathbf{x} \right\| V_j^{-1} \leq \max_{\mathbf{x} \in A_j} \sum_{s=t_{j-1}}^{t_j-1} \left\| \mathbf{x}_s \right\| V_s^{-1} \]
\[ \leq \sum_{s=t_{j-1}}^{t_j-1} \max_{\mathbf{x} \in A_{j-1}} \left\| \mathbf{x}_s \right\| V_s^{-1} \leq \sum_{s=t_{j-1}}^{t_j-1} \left\| \mathbf{x}_s \right\| V_{s-1}^{-1} \]

since the algorithm selects (during phase \( j - 1 \)) the arms with largest width. \( \square \)