Spectral Bandits for Smooth Graph Functions

Michal Valko MICHAL.VALKO@INRIA.FR
INRIA Lille - Nord Europe, SequeL team, 40 avenue Halley 59650, Villeneuve d'Ascq, France
Rémi Munos REMI.MUNOS@INRIA.FR
INRIA Lille - Nord Europe, SequeL team, France; Microsoft Research New England, Cambridge, MA, USA
Branislav Kveton BRANISLAV.KVETON@TECHNICOLOR.COM
Technicolor Research Center, 735 Emerson St, Palo Alto, CA 94301, USA
Tomáš Kocák TOMAS.KOCAK@INRIA.FR
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.

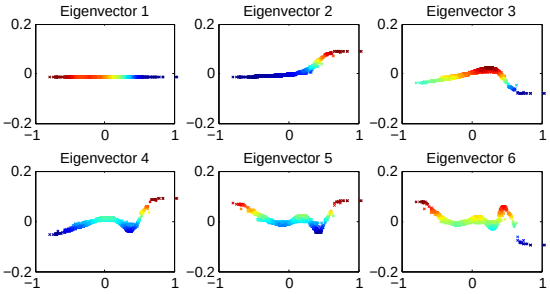

Figure 1. Eigenvectors from the Flixster data corresponding to the smallest few eigenvalues of the graph Laplacian projected onto the first principal component of data. Colors indicate the values.

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

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

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 $\mathcal{G}$ be the given graph with the set of nodes $\mathcal{V}$ and denote $|\mathcal{V}| = N$ the number of nodes. Let $\mathcal{W}$ be the $N \times N$ matrix of similarities w_{ij} (edge weights) and $\mathcal{D}$ is the $N \times N$ diagonal matrix with the entries $d_{ii} = \sum_j w_{ij}$. The graph Laplacian of $\mathcal{G}$ is defined as $\mathcal{L} = \mathcal{D} - \mathcal{W}$. Let $\{\lambda_k^{\mathcal{L}}, \mathbf{q}_k\}_{k=1}^N$ be the eigenvalues and eigenvectors of $\mathcal{L}$ ordered such that $0 = \lambda_1^{\mathcal{L}} \leq \lambda_2^{\mathcal{L}} \leq \cdots \leq \lambda_N^{\mathcal{L}}$. Equivalently, let $\mathcal{L} = \mathbf{Q}\mathbf{\Lambda}_{\mathcal{L}}\mathbf{Q}^\intercal$ be the eigendecomposition of $\mathcal{L}$, where $\mathbf{Q}$ 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 $\boldsymbol{\alpha}$ let $f_{\boldsymbol{\alpha}} : \mathcal{V} \to \mathbb{R}$ be the function defined on nodes, linear in the basis of the eigenvectors of $\mathcal{L}$:

$$f_{\boldsymbol{\alpha}}(v) = \sum_{k=1}^{N} \alpha_k (\mathbf{q}_k)_v = \langle \mathbf{x}_v, \boldsymbol{\alpha} \rangle,$$

where $\mathbf{x}_v$ is the v-th row of $\mathbf{Q}$, i.e., $(\mathbf{x}_v)_i = (\mathbf{q}_i)_v$. If the weight coefficients of the true $\boldsymbol{\alpha}^*$ are such that the large coefficients correspond to the eigenvectors with the small eigenvalues and vice versa, then $f_{\boldsymbol{\alpha}^*}$ would be a smooth function on $\mathcal{G}$ (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 $\boldsymbol{\alpha}^*$ may be of the form $[\alpha_1^*, \alpha_2^*, \ldots, \alpha_k^*, 0, 0, 0]_N^\intercal$ 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 π chooses a node $\pi(t)$ and obtains a noisy reward such that:

$$r_t = \langle \mathbf{x}_{\pi(t)}, \boldsymbol{\alpha}^* \rangle + \varepsilon_t,$$

where the noise ε_t is assumed to be R-sub-Gaussian for any t. In our setting, we have $\mathbf{x}_v \in \mathbb{R}^D$ and $\|\mathbf{x}_v\|_2 \leq 1$ for all $\mathbf{x}_v$. 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. $\boldsymbol{\alpha}^*$. Let $\pi(t)$ be the node picked (referred to as *pulling an arm*) by an algorithm π at time t. The cumulative (pseudo) regret of π is defined as:

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

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 $\sqrt{DT}$ the 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 parameter. We could think of our setting as an N–armed bandit problem where N is larger than the time horizon T and the mean reward μ_k for each arm k satisfies the property that under a change of coordinates, the vector has small weights, i.e., there exists a known orthogonal matrix $\mathbf{U}$ such that $\boldsymbol{\alpha} = \mathbf{U}\boldsymbol{\mu}$ has a low norm. As a consequence, we can estimate $\boldsymbol{\alpha}$ using penalization and then recover $\boldsymbol{\mu}$.

Given a vector of weights $\boldsymbol{\alpha}$, we define its $\boldsymbol{\Lambda}$ norm as:

$$\|\boldsymbol{\alpha}\|_{\boldsymbol{\Lambda}} = \sqrt{\sum_{k=1}^{N} \lambda_k \alpha_k^2} = \sqrt{\boldsymbol{\alpha}^\intercal \boldsymbol{\Lambda} \boldsymbol{\alpha}} \tag{1}$$

We make use of this norm later in our algorithms. Intuitively, we would like to penalize the coefficients $\boldsymbol{\alpha}$ that correspond to the eigenvectors with the large eigenvalues, in other words, to the less smooth basis functions on $\mathcal{G}$.

4.1. Effective dimension

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 $\boldsymbol{\Lambda}$ with the entries $0 < \lambda = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$ and a set of K vectors $\mathbf{x}_1, \ldots, \mathbf{x}_K \in \mathbb{R}^N$ such that $\|\mathbf{x}_i\|_2 \leq 1$ for all i. For the spectral bandits, we have $K = N$. Moreover, since $\mathbf{Q}$ is an orthonormal matrix, $\|\mathbf{x}_i\|_2 = 1$. Finally, since the first eigenvalue of a graph Laplacian is always zero, $\lambda_1^{\mathcal{L}} = 0$, we use $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_{\mathcal{L}} + \lambda\mathbf{I}$, 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 λ_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.

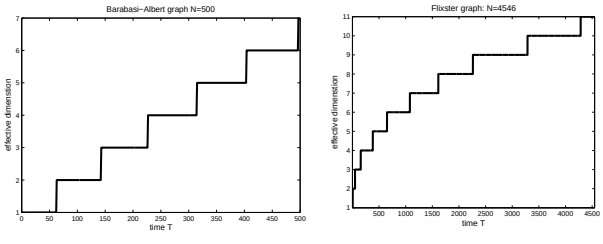

Figure 2. Effective dimension d in the regime $T < N$. The effective dimension for this data is much smaller than the ambient dimension $D = N$, which is 500 and 4546 respectively.

The actual form of Definition 1 comes from Lemma 6 and will become apparent in Section 5. The dependence of the effective dimension on T comes from the fact, that d is related to the number of "non-negligible" dimensions characterizing the space where the solution to the penalized least-squares may lie, since this solution is basically constrained to an ellipsoid defined by the inverse of the eigenvalues multiplied by T. Consequently, d is related to the metric dimension of this ellipsoid. Therefore, when T goes to infinity, the constraint is relaxed and all directions matter, thus the solution can be anywhere in a (bounded) space of dimension N. On the contrary, for a smaller T, the ellipsoid possesses a smaller number of "non-negligible" dimensions. Notice that it is natural that this effective dimension depends on T as we consider the setting $T < N$. If we wanted to avoid T in the definition of d, we could define it as well in terms of N by replacing T by N in Definition 1, but this would only loosen its value.

[1] We set $\boldsymbol{\Lambda}$ to $\boldsymbol{\Lambda}_{\mathcal{L}} + \lambda\mathbf{I}$ with $\lambda = 0.01$, where $\boldsymbol{\Lambda}_{\mathcal{L}}$ is the graph Laplacian of the respective graph.

Algorithm 1 SPECTRALUCB

Input:
N : the number of nodes, T : the number of pulls
$\{\mathbf{\Lambda}_{\mathcal{L}}, \mathbf{Q}\}$ spectral basis of $\mathcal{L}$
λ, δ : regularization and confidence parameters
R, C : upper bounds on the noise and $\|\boldsymbol{\alpha}^*\|_{\mathbf{\Lambda}}$
Run:
$\mathbf{\Lambda} \leftarrow \mathbf{\Lambda}_{\mathcal{L}} + \lambda \mathbf{I}$
$d \leftarrow \max\{d : (d-1)\lambda_d \leq T/\log(1+T/\lambda)\}$
for $t = 1$ **to** T **do**
  Update the basis coefficients $\hat{\boldsymbol{\alpha}}$:
    $\mathbf{X}_t \leftarrow [\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}]^\intercal$
    $r \leftarrow [r_1, \ldots, r_{t-1}]^\intercal$
    $\mathbf{V}_t \leftarrow \mathbf{X}_t \mathbf{X}_t^\intercal + \mathbf{\Lambda}$
    $\hat{\boldsymbol{\alpha}}_t \leftarrow \mathbf{V}_t^{-1} \mathbf{X}_t^\intercal r$
  $c_t \leftarrow 2R\sqrt{d\log(1+t/\lambda) + 2\log(1/\delta)} + C$
  Choose the node v_t ($\mathbf{x}_{v_t}$-th row of $\mathbf{Q}$):
    $v_t \leftarrow \arg\max_v \left(f_{\hat{\boldsymbol{\alpha}}}(v) + c_t \|\mathbf{x}_v\|_{\mathbf{V}_t^{-1}} \right)$
  Observe the reward r_t
end for

Algorithm 2 SPECTRALELIMINATOR

Input:
N : the number of nodes, T : the number of pulls
$\{\mathbf{\Lambda}_{\mathcal{L}}, \mathbf{Q}\}$ spectral basis of $\mathcal{L}$
λ : regularization parameter
$\beta, \{t_j\}_j^J$ parameters of the elimination and phases
$A_1 \leftarrow \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$.
for $j = 1$ **to** J **do**
  $\mathbf{V}_{t_j} \leftarrow \gamma \mathbf{\Lambda}_{\mathcal{L}} + \lambda \mathbf{I}$
  for $t = t_j$ **to** $\min(t_{j+1} - 1, T)$ **do**
    Play $\mathbf{x}_t \in A_j$ with the largest width to observe r_t:
      $\mathbf{x}_t \leftarrow \arg\max_{\mathbf{x} \in A_j} \|\mathbf{x}\|_{\mathbf{V}_t^{-1}}$
    $\mathbf{V}_{t+1} \leftarrow \mathbf{V}_t + \mathbf{x}_t \mathbf{x}_t^\intercal$
  end for
  Eliminate the arms that are not promising:
    $\hat{\boldsymbol{\alpha}}_t \leftarrow \mathbf{V}_t^{-1} [\mathbf{x}_{t_j}, \ldots, \mathbf{x}_t][r_{t_j}, \ldots, r_t]^\intercal$
    $p \leftarrow \max_{\mathbf{x} \in A_j} \left[\langle \hat{\boldsymbol{\alpha}}_t, \mathbf{x} \rangle - \|\mathbf{x}\|_{\mathbf{V}_t^{-1}} \beta \right]$
    $A_{j+1} \leftarrow \left\{ \mathbf{x} \in A_j, \langle \hat{\boldsymbol{\alpha}}_t, \mathbf{x} \rangle + \|\mathbf{x}\|_{V_t^{-1}} \beta \geq p \right\}$
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 $\mathbf{x}_t = \mathbf{x}_{v_t} = \mathbf{x}_{\pi(t)}$. Here we consider regularized least-squares estimate $\hat{\boldsymbol{\alpha}}_t$ of the form:

$$\hat{\boldsymbol{\alpha}}_t = \arg\min_{\boldsymbol{\alpha}} \left(\sum_{v=1}^{t} \left[\langle \mathbf{x}_v, \boldsymbol{\alpha} \rangle - r_v \right]^2 + \|\boldsymbol{\alpha}\|_{\mathbf{\Lambda}} \right)$$

A key part of the algorithm is to define the $c_t \|\mathbf{x}\|_{\mathbf{V}_t^{-1}}$ confidence widths for the prediction 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 λ be the minimum eigenvalue of $\mathbf{\Lambda}$. If $\|\boldsymbol{\alpha}^*\|_{\mathbf{\Lambda}} \leq C$ and for all $\mathbf{x}_v$, $\langle \mathbf{x}_v, \boldsymbol{\alpha}^* \rangle \in [-1, 1]$, then the cumulative regret of* SPECTRALUCB *is with probability at least $1 - \delta$ bounded as:*

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

Remark 1. *The constant C needs to be such that $\|\boldsymbol{\alpha}^*\|_{\mathbf{\Lambda}} \leq C$. If we set C too small, the true $\boldsymbol{\alpha}^*$ will lie outside of the region and far from $\hat{\boldsymbol{\alpha}}_t$, 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 $\|\boldsymbol{\alpha}^*\|_{\mathbf{\Lambda}}$ in advance, but our regret bound would only hold after some t, when $C_t \geq \|\boldsymbol{\alpha}^*\|_{\mathbf{\Lambda}}$.*

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 β. 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.

Theorem 2. *Choose the phases starts as $t_j = 2^{j-1}$. Assume all rewards are in $[0, 1]$ and $\|\boldsymbol{\alpha}^*\|_{\Lambda} \leq C$. For any $\delta > 0$, with probability at least $1 - \delta$, the cumulative regret*

of SPECTRALELIMINATOR *algorithm run with parameter* $\beta = 2R\sqrt{14\log(2K\log_2 T/\delta)} + C$ *is bounded as:*

$$R_T \leq \frac{4}{\log 2}\left(2R\sqrt{14\log\frac{2K\log_2 T}{\delta}}+C\right)\sqrt{dT\log\left(1+\frac{T}{\lambda}\right)}$$

4.4. Scalability and computational complexity

There are three main computational issues to address in order to make SPECTRALUCB 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 Koutis et al. (2011). Furthermore, using CMG we can find good approximations to the first J eigenvectors in $\mathcal{O}(Jm\log m)$ 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 $J = N$ but from our experience, a smooth reward function can be often approximated by dozens of eigenvectors. In fact, J can be considered as an upper bound on the number of eigenvectors we actually need. Furthermore, by choosing small J 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 J can significantly reduce the computation but it is important to choose J 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 J such that our method is able to run. In Section 6.4, we demonstrate that we can obtain good results with relatively small J.

5. Analysis

The analysis of SPECTRALUCB (Section 5.3) has two main ingredients. The first one is the derivation of the confidence ellipsoid for $\hat{\boldsymbol{\alpha}}$, 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 upperbound the regret of SPECTRALELIMINATOR (Section 5.4). The proofs of the lemmas are in the appendix.

5.1. Confidence ellipsoid

The first two lemmas are by Abbasi-Yadkori et al. (2011) and we restate them for the convenience.

Lemma 1. *Let* $\mathbf{V}_t = \mathbf{X}_{t-1}\mathbf{X}_{t-1}^\intercal + \boldsymbol{\Lambda}$ *and define* $\boldsymbol{\xi}_t = \sum_{s=1}^t \varepsilon_s \mathbf{x}_s$. *With probability at least* $1-\delta$, $\forall t \geq 1$*:*

$$\|\boldsymbol{\xi}_t\|^2_{\mathbf{V}_t^{-1}} \leq 2R^2\log\left(\frac{|\mathbf{V}_t|^{1/2}}{\delta|\boldsymbol{\Lambda}|^{1/2}}\right)$$

Lemma 2. *We have:*

$$\sum_{s=1}^t \min\left(1, \|\mathbf{x}_s\|^2_{\mathbf{V}_{s-1}^{-1}}\right) \leq 2\log\frac{|\mathbf{V}_t|}{|\boldsymbol{\Lambda}|}$$

The next lemma is a generalization of Theorem 2 by Abbasi-Yadkori et al. (2011) to the regularization with $\boldsymbol{\Lambda}$. 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 3. *Let* $\mathbf{V}_t = \mathbf{X}_{t-1}\mathbf{X}_{t-1}^\intercal + \boldsymbol{\Lambda}$ *and* $\|\boldsymbol{\alpha}^*\|_{\boldsymbol{\Lambda}} \leq C$. *For any* $\mathbf{x}$ *and* $t \geq 1$*, with probability at least* $1-\delta$*:*

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

5.2. Effective dimension

In Section 5.1 we show that several quantities scale with $\log(|\mathbf{V}_T|/|\boldsymbol{\Lambda}|)$, 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 $\log(|\mathbf{V}_T|/|\boldsymbol{\Lambda}|)$ 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* $\boldsymbol{\Lambda} = diag(\lambda_1, \ldots, \lambda_N)$ *be any diagonal matrix with strictly positive entries and for any vectors* $(\mathbf{x}_t)_{1\leq t\leq T}$ *such that* $\|\mathbf{x}_t\|_2 \leq 1$ *for all* $1 \leq t \leq T$*, we have that the determinant* $|\mathbf{V}_T|$ *of* $\mathbf{V}_T = \boldsymbol{\Lambda} + \sum_{t=1}^T \mathbf{x}_t\mathbf{x}_t^\intercal$ *is maximized when all* $\mathbf{x}_t$ *are aligned with the axes.*

Lemma 5. *For any* T*, let* $\mathbf{V}_T = \sum_{t=1}^T \mathbf{x}_t\mathbf{x}_t^\intercal + \boldsymbol{\Lambda}$. *Then:*

$$\log\frac{|\mathbf{V}_T|}{|\boldsymbol{\Lambda}|} \leq \max\sum_{i=1}^N \log\left(1+\frac{t_i}{\lambda_i}\right),$$

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

Lemma 6. *Let* d *be the effective dimension. Then:*

$$\log\frac{|\mathbf{V}_T|}{|\boldsymbol{\Lambda}|} \leq 2d\log\left(1+\frac{T}{\lambda}\right)$$

5.3. Cumulative regret of SPECTRALUCB

Proof of Theorem 1. Let $\mathbf{x}_* = \arg\max_{\mathbf{x}_v} \mathbf{x}_v^\intercal \boldsymbol{\alpha}^*$ and let $\bar{r}_t$ denote the instantaneous regret at time t. With probability at least $1-\delta$, for all t:

$$\begin{aligned}
\bar{r}_t &= \mathbf{x}_*^\intercal \boldsymbol{\alpha}^* - \mathbf{x}_t^\intercal \boldsymbol{\alpha}^* \\
&\le \mathbf{x}_t^\intercal \hat{\boldsymbol{\alpha}}_t + c_t \|\mathbf{x}_t\|_{\mathbf{V}_t^{-1}} - \mathbf{x}_t^\intercal \boldsymbol{\alpha}^* \qquad (2)\\
&\le \mathbf{x}_t^\intercal \hat{\boldsymbol{\alpha}}_t + c_t \|\mathbf{x}_t\|_{\mathbf{V}_t^{-1}} - \mathbf{x}_t^\intercal \hat{\boldsymbol{\alpha}}_t + c_t \|\mathbf{x}_t\|_{\mathbf{V}_t^{-1}} \qquad (3)\\
&= 2c_t \|\mathbf{x}_t\|_{\mathbf{V}_t^{-1}}.
\end{aligned}$$

The inequality (2) is by the algorithm design and reflects the optimistic principle of SPECTRALUCB. Specifically, $\mathbf{x}_*^\intercal \boldsymbol{\alpha}^* + c_t \|\mathbf{x}_*\|_{\mathbf{V}_t^{-1}} \le \mathbf{x}_t^\intercal \hat{\boldsymbol{\alpha}}_t + c_t \|\mathbf{x}_t\|_{\mathbf{V}_t^{-1}}$, from which:

$$\begin{aligned}
\mathbf{x}_*^\intercal \boldsymbol{\alpha}^* &\le \mathbf{x}_t^\intercal \hat{\boldsymbol{\alpha}}_t + c_t \|\mathbf{x}_t\|_{\mathbf{V}_t^{-1}} - c_t \|\mathbf{x}_*\|_{\mathbf{V}_t^{-1}} \\
&\le \mathbf{x}_t^\intercal \hat{\boldsymbol{\alpha}}_t + c_t \|\mathbf{x}_t\|_{\mathbf{V}_t^{-1}}
\end{aligned}$$

In (3) we applied Lemma 3: $\mathbf{x}_t^\intercal \hat{\boldsymbol{\alpha}}_t \le \mathbf{x}_t^\intercal \boldsymbol{\alpha}^* + c_t \|\mathbf{x}_t\|_{\mathbf{V}_t^{-1}}$. Finally, by Lemmas 2 and 6 and the assumption $\bar{r}_t \le 2$:

$$\begin{aligned}
R_T = \sum_{t=1}^T \bar{r}_t &\le \sqrt{T \sum_{t=1}^T \bar{r}_t^2} \\
&\le 2\max(1, c_T) \sqrt{T \sum_{t=1}^T \min\left(1, \|\mathbf{x}_t\|^2_{\mathbf{V}_t^{-1}}\right)} \\
&\le 2(c_T + 1) \sqrt{2T \log(|\mathbf{V}_T|/|\boldsymbol{\Lambda}|)} \\
&\le 4(c_T + 1) \sqrt{dT \log(1 + T/\lambda)}
\end{aligned}$$

Above, we used that $c_t \le c_T$ because c_t is nondecreasing. By plugging c_T, we get that with probability at least $1-\delta$:

$$\begin{aligned}
R_T &\le 4\left[R\sqrt{2\log(|\mathbf{V}_T|/|\boldsymbol{\Lambda}|) + 2\log(1/\delta)} + C + 1\right] \\
&\quad \times \sqrt{dT \log(1 + T/\lambda)} \\
&\le \left[8R\sqrt{d\log(1 + T/\lambda) + 2\log(1/\delta)} + 4C + 4\right] \\
&\quad \times \sqrt{dT \log(1 + T/\lambda)}
\end{aligned}$$

□

Remark 2. *Notice that if we set $\boldsymbol{\Lambda} = \mathbf{I}$ in Algorithm 1, we recover LinUCB. Since $\log(|\mathbf{V}_T|/|\boldsymbol{\Lambda}|)$ can be upper-bounded by $D \log T$ (Abbasi-Yadkori et al., 2011), we obtain $\tilde{\mathcal{O}}(D\sqrt{T})$ upper bound of regret of LinUCB as a corollary of Theorem 1.*

5.4. Cumulative regret of SPECTRALELIMINATOR

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 $\mathcal{F}_j$ the σ-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 three lemmas for any phase j. Let us write $\mathbf{V}_j$ for $\mathbf{V}_{t_j}$ and $\hat{\boldsymbol{\alpha}}_j$ for $\hat{\boldsymbol{\alpha}}_{t_j}$.

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

$$\mathbb{P}\Big(|\mathbf{x}^\intercal(\hat{\boldsymbol{\alpha}}_j - \boldsymbol{\alpha}^*)| \le \|\mathbf{x}\|_{\mathbf{V}_j^{-1}} \beta(\boldsymbol{\alpha}^*, \delta)\Big) \ge 1 - \delta$$

Lemma 8. *For all $\mathbf{x} \in A_j$, we have:*

$$\|\mathbf{x}\|^2_{\mathbf{V}_j^{-1}} \le \frac{1}{t_j - t_{j-1}} \sum_{s=t_{j-1}+1}^{t_j} \|\mathbf{x}_s\|^2_{\mathbf{V}_{s-1}^{-1}}$$

Lemma 9. *For each phase j, we have:*

$$\sum_{s=t_{j-1}+1}^{t_j} \min\left(1, \|\mathbf{x}_s\|^2_{\mathbf{V}_{s-1}^{-1}}\right) \le \log \frac{|\mathbf{V}_j|}{|\boldsymbol{\Lambda}|}$$

Combining all the lemmas above together we obtain:

Lemma 10. *For each j, $\mathbf{x}$, and δ, with probability at least $1-\delta$:*

$$\begin{aligned}
&\min\left(1, |\mathbf{x}^\intercal(\hat{\boldsymbol{\alpha}}_j - \boldsymbol{\alpha}^*)|\right) \\
&\qquad \le \beta(\boldsymbol{\alpha}^*, \delta) \sqrt{\frac{1}{t_j - t_{j-1}}} \sqrt{2d \log\left(1 + \frac{t_j - t_{j-1}}{\lambda}\right)}
\end{aligned}$$

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

Proof of Theorem 2. Let $J = \lfloor \log_2 T \rfloor$. We have:

$$\begin{aligned}
R_T &= \sum_{t=1}^n \langle \mathbf{x}^* - \mathbf{x}_t, \boldsymbol{\alpha}^* \rangle = \sum_{j=0}^J \sum_{t=t_j}^{t_{j+1}-1} \langle \mathbf{x}^* - \mathbf{x}_t, \boldsymbol{\alpha}^* \rangle = \\
&\le \sum_{j=0}^J (t_{j+1} - t_j) \big[\langle \mathbf{x}^* - \mathbf{x}_t, \hat{\boldsymbol{\alpha}}_j \rangle \\
&\qquad + (\|\mathbf{x}^*\|_{\mathbf{V}_j^{-1}} + \|\mathbf{x}_t\|_{\mathbf{V}_j^{-1}}) \beta \big],
\end{aligned}$$

in an event Ω of probability $1 - KJ\delta$, where we used Lemma 7 in the last inequality. By definition of the action subset A_j at phase j, under Ω, we have:

$$\langle \mathbf{x}^* - \mathbf{x}_t, \hat{\boldsymbol{\alpha}}_j \rangle \le (\|\mathbf{x}^*\|_{\mathbf{V}_j^{-1}} + \|\mathbf{x}_t\|_{\mathbf{V}_j^{-1}}) \beta,$$

since $\mathbf{x}^* \in A_j$ for all $j \le J$. By previous lemma:

$$\begin{aligned}
R_T &\le 2 \sum_{j=0}^J (t_{j+1} - t_j) \beta(\boldsymbol{\alpha}^*, \delta) \sqrt{\frac{1}{t_j - t_{j-1}}} \\
&\qquad \times \sqrt{2d \log\left(1 + \frac{t_j - t_{j-1}}{\lambda}\right)} \\
&\le \frac{4}{\log 2} \beta \sqrt{dT \log\left(1 + \frac{T}{\lambda}\right)}
\end{aligned}$$

□

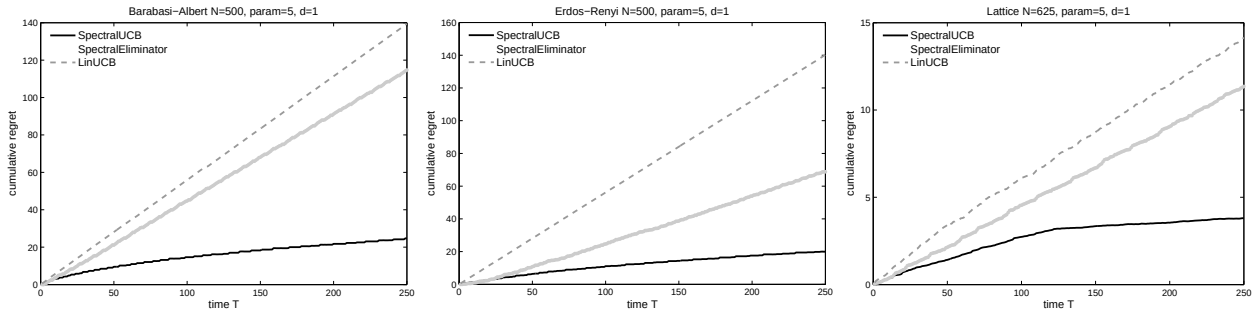

Figure 3. Cumulative regret for random graphs models

Remark 3. *If we set $\mathbf{\Lambda} = \mathbf{I}$ 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 $\tilde{\mathcal{O}}(\sqrt{DT})$ 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 δ to 0.001 and R to 0.01. For SPECTRALUCB and SPECTRALELIMINATOR we set $\mathbf{\Lambda}$ to $\mathbf{\Lambda}_{\mathcal{L}} + \lambda\mathbf{I}$ 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[2], but due to the known reasons (Chu et al., 2011) these algorithms are not efficient[3] 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 $\boldsymbol{\alpha}^*$ of N weights, $k \ll N$ and defined the true graph function as $f = \mathbf{Q}\boldsymbol{\alpha}^*$, 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[4] 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 $\mathbf{M} \approx \mathbf{U}\mathbf{V}'$, 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.

[2] an equivalent of SupLinUCB with spectral regularization

[3] at least for the sizes of N and T that we deal with

[4] For example, these algorithms do not to use all the rewards obtained for the estimation of $\hat{\boldsymbol{\alpha}}$.

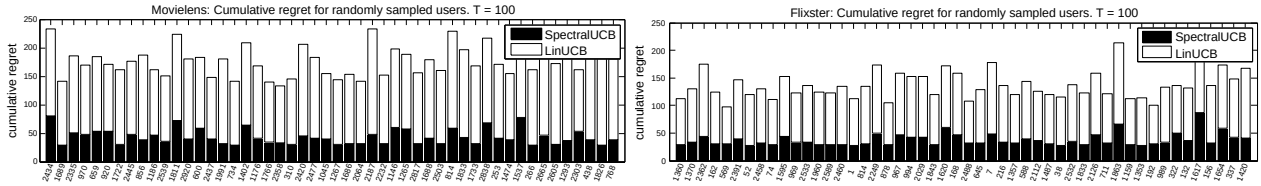

Figure 4. MovieLens and Flixter: Cumulative regret for 50 randomly chosen users. Horizontal axis shows the user number.

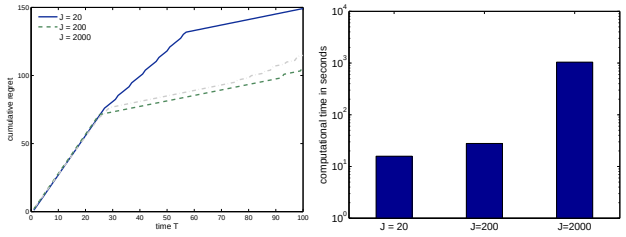

Figure 5. Regret and computational time with reduced basis

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, 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 $J \ll N$ eigenvectors of the graph Laplacian. First, the computational complexity of such operation is $\mathcal{O}(Jm \log m)$, 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 J 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 J 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, SPECTRALUCB and 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 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 Flixster 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.

References

Abbasi-Yadkori, Y, Pál, D, and Szepesvári, C. Improved Algorithms for Linear Stochastic Bandits. In *Neural Information Processing Systems*. 2011.

Abernethy, J. D, Hazan, E, and Rakhlin, A. Competing in the Dark: An Efficient Algorithm for Bandit Linear Optimization. In *Conference on Learning Theory*, 2008.

Alon, N, Cesa-Bianchi, N, Gentile, C, and Mansour, Y. From Bandits to Experts: A Tale of Domination and Independence. In *NIPS*, 2013.

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, March 2002. ISSN 1532-4435.

Auer, P and Ortner, R. UCB Revisited: Improved Regret Bounds for the Stochastic Multi-Armed Bandit Problem. *Periodica Mathematica Hungarica*, 2010.

Azuma, K. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 19(3):357–367, 1967.

Barabási, A.-L and Albert, R. Emergence of scaling in random networks. *Science*, 286:11, 1999.

Belkin, M, Matveeva, I, and Niyogi, P. Regularization and Semi-Supervised Learning on Large Graphs. In *Conference on Learning Theory*, 2004.

Belkin, M, Niyogi, P, and Sindhwani, V. Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

Billsus, D, Pazzani, M. J, and Chen, J. A learning agent for wireless news access. In *IUI*, pp. 33–36, 2000.

Bubeck, S, Munos, R, Stoltz, G, and Szepesvari, C. X-armed bandits. *Journal of Machine Learning Research*, 12:1587–1627, 2011.

Bubeck, S, Cesa-Bianchi, N, and Kakade, S. Towards minimax policies for online linear optimization with bandit feedback. In *COLT*, 2012.

Caron, S, Kveton, B, Lelarge, M, and Bhagat, S. Leveraging Side Observations in Stochastic Bandits. In *Uncertainty in Artificial Intelligence*, pp. 142–151, 2012.

Cesa-Bianchi, N, Gentile, C, and Zappella, G. A Gang of Bandits. In *NIPS*, 2013.

Chau, D. H, Kittur, A, Hong, J. I, and Faloutsos, C. Apolo: making sense of large network data by combining rich user interaction and machine learning. In *CHI*, 2011.

Chu, L, Li, L, Reyzin, L, and Schapire, R. Contextual Bandits with Linear Payoff Functions. In *AISTATS*, 2011.

Dani, V, Hayes, T. P, and Kakade, S. M. Stochastic Linear Optimization under Bandit Feedback. In *The 21st Annual Conference on Learning Theory*, 2008.

Desautels, T, Krause, A, and Burdick, J. Parallelizing Exploration-Exploitation Tradeoffs with Gaussian Process Bandit Optimization. In *ICML*, 2012.

Graclus. Graclus, 2013. URL www.cs.utexas.edu/users/dml/Software/graclus.html.

Jamali, M and Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010.

Jannach, D, Zanker, M, Felfernig, A, and Friedrich, G. *Recommender Systems: An Introduction*. Cambridge University Press, 2010.

Keshavan, R, Oh, S, and Montanari, A. Matrix Completion from a Few Entries. In *IEEE International Symposium on Information Theory*, pp. 324–328, 2009.

Kleinberg, R, Slivkins, A, and Upfal, E. Multi-armed bandit problems in metric spaces. In *40th ACM symposium on Theory Of Computing*, 2008.

Koutis, I, Miller, G. L, and Tolliver, D. Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. *Computer Vision and Image Understanding*, 115:1638–1646, 2011.

Lam, S and Herlocker, J. MovieLens 1M Dataset. http://www.grouplens.org/node/12, 2012.

Li, L, Chu, W, Langford, J, and Schapire, R. E. A Contextual-Bandit Approach to Personalized News Article Recommendation. *WWW 10*, 2010.

McPherson, M, Smith-Lovin, L, and Cook, J. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27:415–444, 2001.

Slivkins, A. Contextual Bandits with Similarity Information. *Proceedings of the 24th annual Conference On Learning Theory*, pp. 1–27, 2009.

Srinivas, N, Krause, A, Kakade, S, and Seeger, M. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *Proceedings of International Conference on Machine Learning*, 2010.

Valko, M, Korda, N, Munos, R, Flaounas, I, and Cristianini, N. Finite-Time Analysis of Kernelised Contextual Bandits. In *Uncertainty in Artificial Intelligence*, 2013.

Zhang, F. *The Schur complement and its applications*, volume 4. Springer, 2005.

Zhu, X. Semi-Supervised Learning Literature Survey. Technical Report 1530, U. of Wisconsin-Madison, 2008.