Spectral Bandits for Smooth Graph Functions
with Applications in Recommender Systems

Tomáš Kocák
SequeL team
INRIA Lille
France

Michal Valko
SequeL team
INRIA Lille
France

Rémi Munos
SequeL team, INRIA France
Microsoft Research
New England

Branislav Kveton
Technicolor
Research Center
California

Shipra Agrawal
Microsoft Research
Bangalore
India

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 recommended item 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 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 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 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 solutions on the neighboring nodes tend to be similar. It is well-known (Belkin, Niyogi, and Sindhwani, 2006; Belkin, Matveeva, and Niyogi, 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 as a 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 for pulling an arm is a smooth function on this graph.

Our work is motivated by a range of practical problems that involve graphs. One potential application is *targeted advertising* in social networks. In this problem, 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 change smoothly (McPherson, Smith-Lovin, and Cook, 2001), because friends tend to have similar preferences. Therefore, we can take advantage of this
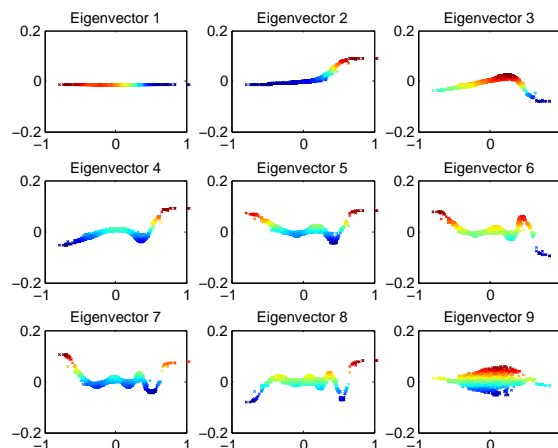
Figure 1: Eigenvectors from the Flixster data corresponding to the smallest few eigenvalues of the graph Laplacian projected onto the first principal component of data. Colors indicate the values.

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 (Pazzani and Billsus, 2007), the user is recommended items that are similar to the items that the user rated highly in the past. The assumption is the user prefers similar items similarly. The similarity of the items can be measured in many ways, for instance by a nearest neighbor graph (Billsus, Pazzani, and Chen, 2000), where each item is a node and its neighbors are most similar items. Therefore, the problem of learning items that the user likes the most, can be naturally formulated in our framework.

In this paper, we consider the following learning setting. The graph is known in advance and its edges represent the similarity of the nodes in the graph. At time t, we choose a node and then observe its payoff. In targeted advertising, 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 we proceed 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 trivially 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, a 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 our problem is nontrivial.

We make several major contributions. First, we formalize a bandit problem, where the payoff of the arms is a smooth function on a graph. Second, we propose two algorithms for solving this problem that achieve $d\sqrt{T \ln T}$ and $d\sqrt{T \ln N}$ expected cumulative regret, where d is the effective dimension of the problem. Finally, we evaluate both of the algorithms on synthetic and real-world content-based recommendation problems.

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 $\mathcal{L} = \mathbf{Q}\mathbf{\Lambda}_{\mathcal{L}}\mathbf{Q}^\mathsf{T}$ 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 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, Niyogi, and Sindhwani, 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 $[\alpha_1^*, \alpha_2^*, \ldots, \alpha_k^*, 0, 0, 0]_N^\mathsf{T}$ 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. 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 norm of $\boldsymbol{\alpha}^*$ which expresses 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 $r_t = \mathbf{x}_{\pi(t)}^\mathsf{T} \boldsymbol{\alpha}^* + \varepsilon_t$, where the noise ε_t is assumed to be R-sub-Gaussian for any t, i.e.,

$$\forall \xi \in \mathbb{R}, \ \mathbb{E}[e^{\xi \varepsilon_t}] \leq \exp\left(\frac{\xi^2 R^2}{2}\right).$$

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 also to the dimension of the basis D (each eigenvector is 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 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 bandits setting. Intuitively, the effective dimension is a proxy for number of relevant dimensions. We first provide a formal definition and then discuss its properties.

In general, we assume there exists a diagonal matrix $\mathbf{\Lambda}$ 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 smooth graph functions, we have $K = N$. Moreover since $\mathbf{Q}$ is an orthonormal matrix, $\|\mathbf{x}_i\|_2 = 1$. Finally, since the first eigenvalues of a graph Laplacian is always zero, $\lambda_1^{\mathcal{L}} = 0$, we use $\mathbf{\Lambda} = \mathbf{\Lambda}_{\mathcal{L}} + \lambda \mathbf{I}$, 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) are much larger than T, such as in graphs from social networks with very large number of nodes N. In the contrary, when the coefficients are all small then d may be of order 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.

---

[1] We set $\mathbf{\Lambda}$ to $\mathbf{\Lambda}_{\mathcal{L}} + \lambda \mathbf{I}$ with $\lambda = 0.01$, where $\mathbf{\Lambda}_{\mathcal{L}}$ is the graph Laplacian of the respective graph.
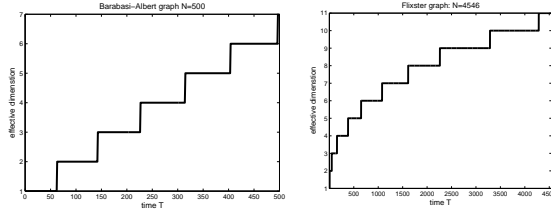
Figure 2: Effective dimension d in the regime $T < N$. The effective dimension for this data is much smaller than the ambient dimension D, which is 500 and 4546 respectively.

Algorithm 1 SpectralUCB

Input:
    N : the number of vertices, 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 basis coefficients $\hat{\boldsymbol{\alpha}}$:
        $\mathbf{X}_t \leftarrow [\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}]^\top$
        $r \leftarrow [r_1, \ldots, r_{t-1}]^\top$
        $\mathbf{V}_t \leftarrow \mathbf{X}_t \mathbf{X}_t^\top + \mathbf{\Lambda}$
        $\hat{\boldsymbol{\alpha}}_t \leftarrow \mathbf{V}_t^{-1} \mathbf{X}_t^\top r$
    $c_t \leftarrow 2R\sqrt{d\log(1 + t/\lambda) + 2\log(1/\delta)} + C$
    Choose 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 reward r_t
end for

---

4 Algorithms

In this section we propose two algorithms. The first algorithm is based on LinUCB. For the second algorithm we use Thompson sampling (TS) approach.

SpectralUCB

The first algorithm we present is SpectralUCB (Algorithm 1) which is based on LinUCB and uses the *spectral penalty*. 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:

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

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 to define c_t based on the effective dimension d which is specifically tailored to our setting. The following theorem characterizes the performance of SpectralUCB and bounds the regret as a function of effective dimension d.

Theorem 1 (Valko et al. (2014)). *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 upperbound on $\|\boldsymbol{\alpha}^*\|_{\mathbf{\Lambda}}$ in advance, but our regret bound would only hold after some t, where $C_t \geq \|\boldsymbol{\alpha}^*\|_{\mathbf{\Lambda}}$.*

Spectral Thompson Sampling

In this section, we use the Thompson Sampling approach to decide which arm to play. Specifically, we will represent our current knowledge about $\boldsymbol{\alpha}^*$ as the normal distribution $\mathcal{N}(\hat{\boldsymbol{\alpha}}(t), v^2 \mathbf{V}_t^{-1})$, where $\hat{\boldsymbol{\alpha}}(t)$ is our actual approximation of the unknown parameter $\boldsymbol{\alpha}^*$ and $v^2 \mathbf{V}_t^{-1}$ reflects of our uncertainty about it for some constant v specifically selected using your analysis (Kocák et al. (2014)). As mentioned before, we assume that the reward function is a linear combination of eigenvectors of $\mathcal{L}$ with large coefficients corresponding to the eigenvectors with small eigenvalues. We encode this assumption into our initial confidence ellipsoid by setting $\mathbf{V}_1 = \mathbf{\Lambda} = \mathbf{\Lambda}_{\mathcal{L}} + \lambda \mathbf{I}$, where λ is a regularization parameter.

After that, every time step t we generate a sample $\tilde{\boldsymbol{\alpha}}(t)$ from distribution $\mathcal{N}(\hat{\boldsymbol{\alpha}}(t), v^2 \mathbf{V}_t^{-1})$ and chose an arm $\pi(t)$, that maximizes $\mathbf{x}_i^\top \tilde{\boldsymbol{\alpha}}(t)$. After receiving a reward, we update our estimate of $\boldsymbol{\alpha}^*$ and the confidence of it, i.e. we compute $\hat{\boldsymbol{\alpha}}(t+1)$ and $\mathbf{V}(t+1)$,

$$\mathbf{V}_{t+1} = \mathbf{V}_t + \mathbf{x}_{\pi(t)} \mathbf{x}_{\pi(t)}^\top$$
$$\hat{\boldsymbol{\alpha}}(t+1) = \mathbf{V}_{t+1}^{-1} \left(\sum_{i=1}^{t} \mathbf{x}_{\pi(i)} r(i) \right).$$

The computational advantage of SpectralTS in Algorithm 2, compared to SpectralUCB, is that we do not need to compute the confidence bound for each arm. Indeed, in SpectralTS we need to sample $\tilde{\boldsymbol{\alpha}}$ which can be done in N^2 time (note that $\mathbf{V}_t$ is only changing by a rank one update) and a maximum of $\mathbf{x}_i^\top \tilde{\boldsymbol{\alpha}}$ which can be also done in N^2 time. On the other hand, in SpectralUCB, we need to compute a $\mathbf{V}_t^{-1}$ norm for each of N context vectors which amounts to a ND^2 time. Table 1 (left) summarizes the computational complexity of the two approaches. Finally note that in our setting $D = N$, which comes to a N^2 vs. N^3 time per step. We support this argument in Section 5. Finally note that the eigendecomposition needs to be done only once in the beginning and since $\mathcal{L}$ is diagonally dominant, this can be done for N in millions (Koutis, Miller, and Peng, 2010).

We would like to stress that we consider the regime when $T < N$, because we aim for applications with a large set of arms and we are interested in a satisfactory performance

	Linear	Spectral
Optimistic Approach D^2N per step update	**LinUCB** $D\sqrt{T\ln T}$	**SpectralUCB** $d\sqrt{T\ln T}$
Thompson Sampling $D^2 + DN$ per step update	**LinearTS** $D\sqrt{T\ln N}$	*SpectralTS* $d\sqrt{T\ln N}$

Table 1: Linear vs. Spectral Bandits

Algorithm 2 Spectral Thompson Sampling

Input:
N: number of arms, T: number of pulls
$\{\Lambda_{\mathcal{L}}, \mathbf{Q}\}$: spectral basis of graph Laplacian $\mathcal{L}$
λ, δ: regularization and confidence parameters
R, C: upper bounds on noise and $\|\boldsymbol{\alpha}^*\|_{\Lambda}$

Initialization:
$v = R\sqrt{6d\ln((\lambda+T)/\delta\lambda)} + C$
$\hat{\boldsymbol{\alpha}} = 0_N, \boldsymbol{f} = 0_N, \mathbf{V} = \Lambda_{\mathcal{L}} + \lambda\mathbf{I}_N$

Run:
for $t = 1$ **to** T **do**
  Sample $\tilde{\boldsymbol{\alpha}} \sim \mathcal{N}(\hat{\boldsymbol{\alpha}}, v^2\mathbf{V}^{-1})$
  $\pi(t) \leftarrow \arg\max_a \mathbf{x}_a^\intercal \tilde{\boldsymbol{\alpha}}$
  Observe a noisy reward $r(t) = \mathbf{x}_{\pi(t)}^\intercal \boldsymbol{\alpha}^* + \varepsilon$
  $\boldsymbol{f} \leftarrow \boldsymbol{f} + \mathbf{x}_{\pi(t)} r(t)$
  Update $\mathbf{V} \leftarrow \mathbf{V} + \mathbf{x}_{\pi(t)}\mathbf{x}_{\pi(t)}^\intercal$
  Update $\hat{\boldsymbol{\alpha}} \leftarrow \mathbf{V}^{-1}\boldsymbol{f}$
end for

---

after just a few iterations. For instance, when we aim to recommend N movies, we would like to have useful recommendations in the time $T < N$, i.e., before the user saw all of them. The following theorem upperbounds the cumulative regret of SpectralTS in terms of d.

Theorem 2 (Kocák et al. (2014)). *Let d be the effective dimension and λ be the minimum eigenvalue of Λ. If $\|\boldsymbol{\alpha}^*\|_{\Lambda} \leq C$ and for all $\mathbf{x}_i$, $|\mathbf{x}_i^\intercal \boldsymbol{\alpha}^*| \leq 1$, then the cumulative regret of Spectral Thompson Sampling is with probability at least $1 - \delta$ bounded as*

$$\mathcal{R}_T \leq \frac{11g}{p}\sqrt{\frac{4+4\lambda}{\lambda}dT\ln\frac{\lambda+T}{\lambda}} + \frac{1}{T}$$
$$+ \frac{g}{p}\left(\frac{11}{\sqrt{\lambda}}+2\right)\sqrt{2T\ln\frac{2}{\delta}},$$

where $p = 1/(4e\sqrt{\pi})$ and

$$g = \sqrt{4\ln TN}\left(R\sqrt{6d\ln\left(\frac{\lambda+T}{\delta\lambda}\right)}+C\right)$$
$$+ R\sqrt{2d\ln\left(\frac{(\lambda+T)T^2}{\delta\lambda}\right)} + C.$$

Remark 2. *Substituting g and p we see that regret bound scales with $d\sqrt{T\ln N}$. Note that $N = D$ could be exponential in d and we need to consider factor $\sqrt{\ln N}$ in our bound. On the other hand if N is exponential in d, then our algorithm scales with $\ln D\sqrt{T\ln D} = \ln(D)^{3/2}\sqrt{T}$ which is even better.*

5 Experiments

The aim of this section is to give empirical evidence that SpectralTS and SpectralUCB deliver better empirical performance than LinearTS and LinUCB in $T < N$ regime. For the synthetic experiment, we considered *Barabási-Albert* (BA) model (1999), known for its preferential attachment property, common in real-world graphs. We generated a random graph using BA model with $N = 250$ nodes and the degree parameter 3. For each run, we generated the weights of the edges uniformly at random. Then we generated $\boldsymbol{\alpha}^*$, a random vector of weights (unknown) to algorithms in order to compute the payoffs and evaluated the cumulative regret. The $\boldsymbol{\alpha}^*$ in each simulation was a random linear combination of the first 3 smoothest eigenvectors of the graph Laplacian. In all experiments, we had $\delta = 0.001$, $\lambda = 1$, and $R = 0.01$. We evaluated the algorithms in $T < N$ regime, where the linear bandit algorithms are not expected to perform well. Figure 3 shows the results averaged over 10 simulations. Notice that while the result of SpectralTS are comparable to SpectralUCB, its computational time is much faster due the reasons discussed in Section 4. Recall that while both algorithms compute the least-square problem of the same size, SpectralUCB has then to compute the confidence interval for each arm.
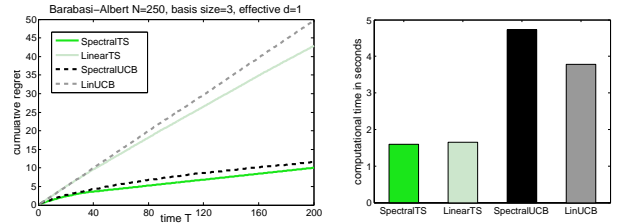

Figure 3: Barabási-Albert random graph results

Furthermore, we performed the comparison on the MovieLens dataset (Lam and Herlocker, 2012) of the movie ratings. The graph in this dataset is the graph of 2019 movies edges corresponding to the movie similarities. We completed the missing ratings by a low-rank matrix factorization and used it construct a 10-NN similarity graph. For each user we have a graph function, unknown to the algorithms, that assigns to each node, the rating of the particular user. A detailed description on the preprocessing is deferred to (Valko et al., 2014) due to limited space. Our goal is then to recommend the most highly rated content. Figure 4 shows the MovieLens data results averaged over 10 randomly sampled users. Notice that the results follow the same trends as for synthetic data. Overall, both our spectral algorithms outperform their linear counterparts in terms of cumulative regret.
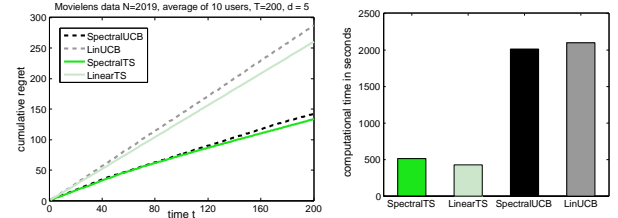

Figure 4: MovieLens data results

References

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

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

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

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

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

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

Kocák, T.; Valko, M.; Munos, R.; and Agrawal, S. 2014. Spectral Thompson Sampling. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Koutis, I.; Miller, G. L.; and Peng, R. 2010. Approaching Optimality for Solving SDD Linear Systems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 235–244. IEEE.

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

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

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

Pazzani, M. J., and Billsus, D. 2007. Content-Based Recommendation Systems. *The adaptive web*.

Valko, M.; Munos, R.; Kveton, B.; and Kocák, T. 2014. Spectral Bandits for Smooth Graph Functions. In *31th International Conference on Machine Learning*.

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