Graphs in Machine Learning

Michal Valko

INRIA Lille - Nord Europe, France

Partially based on material by: Rob Fergus, Tomáš Kocák

Last Lecture

- Analysis of online SSL
- Analysis of quantization error
- When does graph-based SSL provably help?

This Lecture

- Scaling harmonic functions to millions of samples
- Online decision-making on graphs
- Graph bandits
  - smoothness of rewards (preferences) on a given graph
  - observability graphs
  - side information

Previous Lab Session

- 10. 3. 2015 by Daniele.Calandriello@inria.fr
- Content
  - GraphLab
  - Large-Scale Graph Learning
- Short written report (graded, each lab around 5% of grade)
- Questions to Daniele.Calandriello@inria.fr
- *Deadline:* 24. 3. 2015
- `http://researchers.lille.inria.fr/~calandri/ta/graphs/td3_handout.pdf`

Final Class projects

- preferred option: you come up with the topic
- details and list of suggested topics on the class website
- theory/implementation/review or a combination
- one or two people per project (exceptionally three)
- grade: report + short presentation of the team
- Deadlines
  - taking projects 17. 3. 2015 - today
  - 11. 4. 2015 final report
  - 13. 4. 2015 afternoon, presentation in class

http://researchers.lille.inria.fr/~valko/hp/mvaprojects

Ph.D. position in Lille and Amsterdam

PhD position in Theoretical Machine Learning is offered at Inria Lille. Possibility of a joint PhD with CWI, Amsterdam. Lille is 1h away from Paris, 34min from Brussels, 1h30 from London and 2h30 from Amsterdam, all by (fast) train. (And Amsterdam is in Amsterdam.)

The topic is to explore which regularities are "learnable" from data. Specifically, the focus is on the problem of forecasting, that is, predicting the probabilities of future outcomes of a series of events given the past. The question to be addressed is: under which assumptions on the stochastic mechanism generating the data is it possible to construct a consistent forecaster?

The student will be advised by Daniil.Ryabko@inria.fr, to whom all inquiries should be directed. The topic is highly mathematical. Please do not apply if you don't like **proving theorems.**

Scaling SSL with Graphs to Millions

Semi-supervised learning with graphs

$$\mathbf{f}^\star = \min_{\mathbf{f}\in\mathbb{R}^n} (\mathbf{f}-\mathbf{y})^\mathsf{T}\mathbf{C}(\mathbf{f}-\mathbf{y}) + \mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f}$$

Let us see the same in eigenbasis of $\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\mathsf{T}$, i.e., $\mathbf{f} = \mathbf{U}\boldsymbol{\alpha}$

$$\boldsymbol{\alpha}^\star = \min_{\boldsymbol{\alpha}\in\mathbb{R}^n} (\mathbf{U}\boldsymbol{\alpha}-\mathbf{y})^\mathsf{T}\mathbf{C}(\mathbf{U}\boldsymbol{\alpha}-\mathbf{y}) + \boldsymbol{\alpha}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{\alpha}$$

What is the problem with scalability?

Diagonalization of $n \times n$ matrix

What can we do? Let's take only first k eigenvectors $\mathbf{f} = \mathbf{U}\boldsymbol{\alpha}$!

Scaling SSL with Graphs to Millions

$\mathbf{U}$ is now a $n \times k$ matrix

$$\boldsymbol{\alpha}^\star = \min_{\boldsymbol{\alpha}\in\mathbb{R}^n} (\mathbf{U}\boldsymbol{\alpha} - \mathbf{y})^\mathsf{T}\mathbf{C}(\mathbf{U}\boldsymbol{\alpha} - \mathbf{y}) + \boldsymbol{\alpha}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{\alpha}$$

Closed form solution is $(\boldsymbol{\Lambda} + \mathbf{U}^\mathsf{T}\mathbf{C}\mathbf{U})\boldsymbol{\alpha} = \mathbf{U}^\mathsf{T}\mathbf{C}\mathbf{y}$

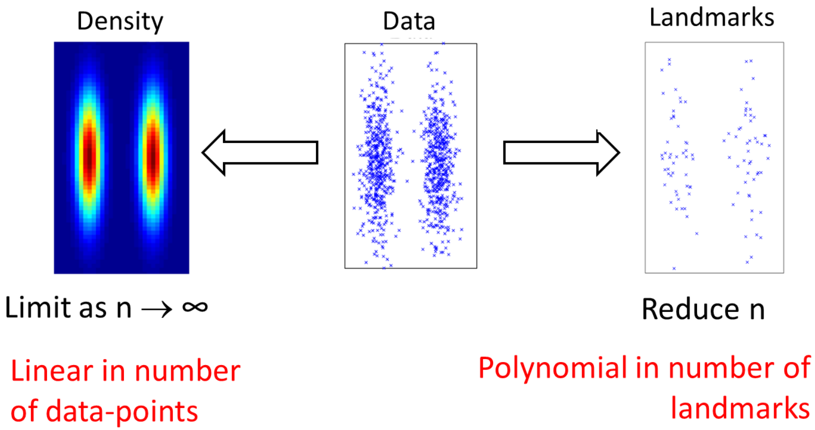
What is the size of this system of equation now?

$k \times k$! Cool! Any problem with this approach?

Getting $\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\mathsf{T}$ is a $n \times n$ system :(

Let's see what happens when $n \to \infty$!

Scaling SSL with Graphs to Millions



https://cs.nyu.edu/~fergus/papers/fwt_ssl.pdf

Scaling SSL with Graphs to Millions

What happens to L when $n \to \infty$?

We have data $\mathbf{x}_i \in \mathbb{R}$ sampled from $p(\mathbf{x})$.

When $n \to \infty$, instead of $\mathbf{f}$ we consider functions $F(x)$.

Instead of $\mathbf{L}$ we define $\mathcal{L}_p$ - **weighted smoothness operator**

$$\mathcal{L}_p(F) = \frac{1}{2}\int (F(\mathbf{x}_1) - F(\mathbf{x}_2))^2 W(\mathbf{x}_1, \mathbf{x}_2) p(\mathbf{x}_1) p(\mathbf{x}_2) \mathrm{d}\mathbf{x}_1 \mathbf{x}_2$$

$$\text{with } W(\mathbf{x}_1, \mathbf{x}_2) = \frac{\exp\left(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2\right)}{2\sigma^2}$$

$\mathbf{L}$ defined the eigenvectors of increasing smoothness.

What defines $\mathcal{L}_p$? **Eigenfunctions!**

Scaling SSL with Graphs to Millions

$$\mathcal{L}_p(F) = \frac{1}{2}\int (F(\mathbf{x}_1) - F(\mathbf{x}_2))^2 W(\mathbf{x}_1, \mathbf{x}_2) p(\mathbf{x}_1) p(\mathbf{x}_2) \, \mathrm{d}x_1 x_2$$

First eigenfunction

$$\Phi_1 = \underset{F : \int F^2(\mathbf{x}) p(\mathbf{x}) D(\mathbf{x}) \, \mathrm{d}x = 1}{\arg\min} \mathcal{L}_p(F)$$

where $D(\mathbf{x}) = \int_{\mathbf{x}_2} W(\mathbf{x}, \mathbf{x}_2) p(\mathbf{x}_2) \, \mathrm{d}\mathbf{x}_2$

What is the solution? $\Phi_1(\mathbf{x}) = 1$ because $\mathcal{L}_p(1) = 0$

How to define Φ_2? Same constraining to be orthogonal to Φ_1

$$\int F(\mathbf{x}) \Phi_1(\mathbf{x}) p(\mathbf{x}) D(\mathbf{x}) \, \mathrm{d}x = 0$$

Scaling SSL with Graphs to Millions

Eigenfunctions of $\mathcal{L}_p$

Φ_3 as before, orthogonal to Φ_1 and Φ_2 etc.

How to define eigenvalues? $\lambda_k = \mathcal{L}_p(\Phi_k)$

Relationship to the discrete Laplacian

$$\frac{1}{n^2}\mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f} = \frac{1}{2n^2}\sum_{ij} W_{ij}(f_i - f_j)^2 \xrightarrow[n\to\infty]{} \mathcal{L}_p(F)$$

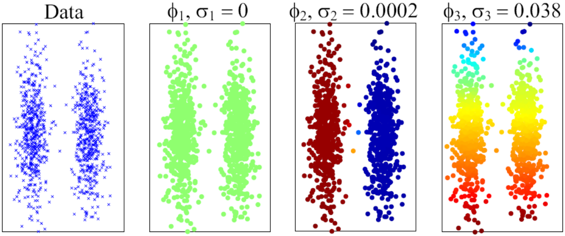
Isn't estimating eigenfunctions $p(\mathbf{x})$ more difficult? Yes it is.

Are there some "easy" distributions?

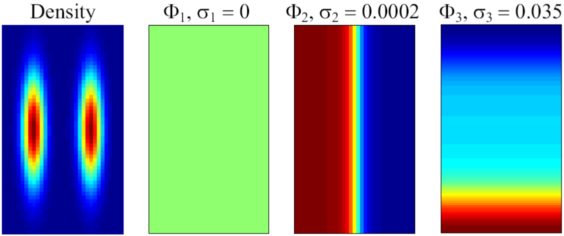
Can we compute is numerically?

Scaling SSL with Graphs to Millions

Eigenvectors



Eigenfunctions

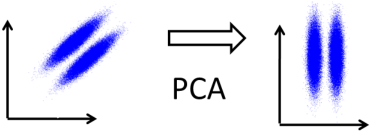


Scaling SSL with Graphs to Millions

Factorized data distribution What if

$$p(\mathbf{s}) = p(s_1)\, p(s_2) \ldots p(s_d)$$

In general this is not true. But we can rotate data with $\mathbf{s} = \mathbf{R}\mathbf{x}$.



Treating each factor individually

$p_k \stackrel{\text{def}}{=}$ marginal distribution of s_k

$\Phi_i(s_k) \stackrel{\text{def}}{=}$ eigenfunction of $\mathcal{L}_{p_k}$ with eigenvalue λ_i

Then: $\Phi_i(s) = \Phi_i(s_k)$ is eigenfunction of $\mathcal{L}_p$ with λ_i

Scaling SSL with Graphs to Millions

How to approximate 1D density? Histograms!

Algorithm of Fergus et al. [FWT09] for eigenfunctions skip

- Find $\mathbf{R}$ such that $\mathbf{s} = \mathbf{Rx}$
- For each "independent" s_k approximate $p(s_k)$
- Given $p(s_k)$ numerically solve for eigensystem of $\mathcal{L}_{p_k}$

$$\left(\tilde{\mathbf{D}} - \mathbf{P}\tilde{\mathbf{W}}\mathbf{P}\right)\mathbf{g} = \lambda \mathbf{P}\widehat{\mathbf{D}}\mathbf{g} \qquad \text{(generalized eigensystem)}$$

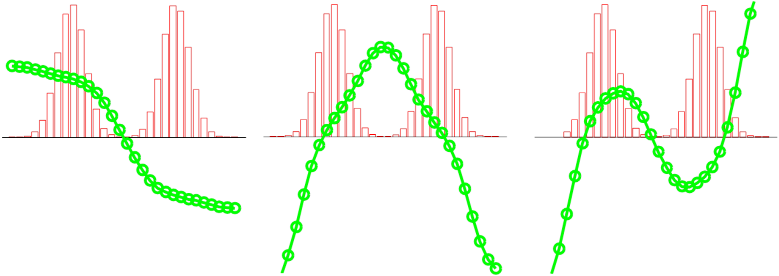
$\mathbf{g}$ - vector of length $B \equiv$ number of bins
$\mathbf{P}$ - density at discrete points
$\tilde{\mathbf{D}}$ - diagonal sum of $\mathbf{P}\tilde{\mathbf{W}}\mathbf{P}$
$\widehat{\mathbf{D}}$ - diagonal sum of $\mathbf{P}\tilde{\mathbf{W}}$

- Order eigenfunctions by increasing eigenvalues

https://cs.nyu.edu/~fergus/papers/fwt_ssl.pdf

Scaling SSL with Graphs to Millions

Numerical 1D Eigenfunctions



1^{st} Eigenfunction of $h(x_1)$ — 2^{nd} Eigenfunction of $h(x_1)$ — 3^{rd} Eigenfunction of $h(x_1)$

https://cs.nyu.edu/~fergus/papers/fwt_ssl.pdf

Scaling SSL with Graphs to Millions

Computational complexity for $n \times d$ dataset

Typical harmonic approach

one diagonalization of $n \times n$ system

Numerical eigenfunctions with B bins and k eigenvectors

d eigenvector problems of $B \times B$

$$\left(\tilde{\mathbf{D}} - \mathbf{P}\tilde{\mathbf{W}}\mathbf{P}\right)\mathbf{g} = \lambda \mathbf{P}\widehat{\mathbf{D}}\mathbf{g}$$

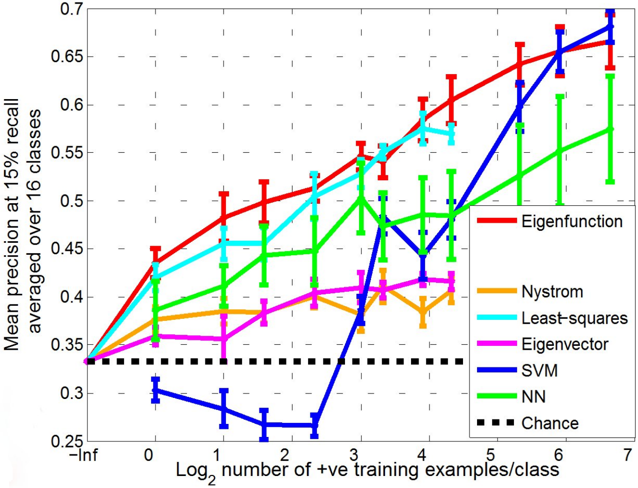
one $k \times k$ least squares problem

$$(\mathbf{\Lambda} + \mathbf{U}^\mathsf{T}\mathbf{C}\mathbf{U})\alpha = \mathbf{U}^\mathsf{T}\mathbf{C}\mathbf{y}$$

some details: several approximation, eigenvectors only linear combinations single-coordinate eigenvectors, ...

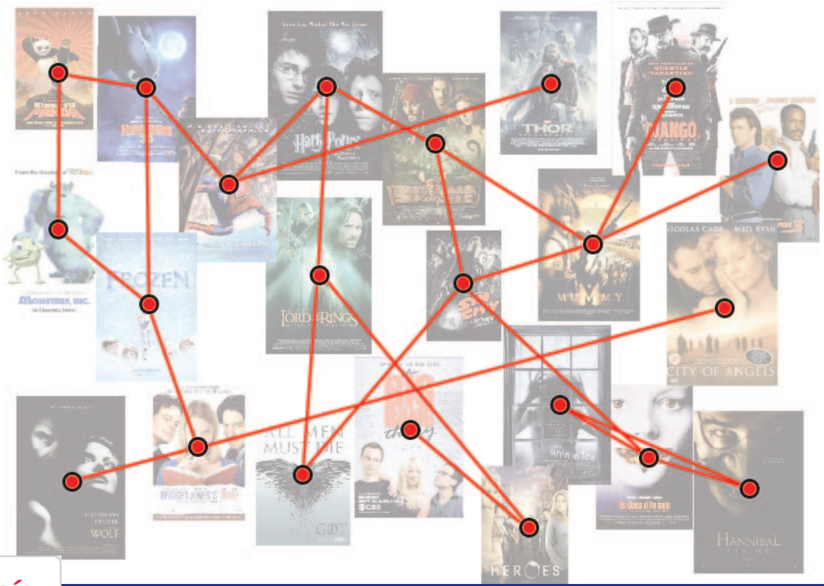
When d is not too big then n can be in millions!

Scaling SSL with Graphs to Millions



CIFAR experiments `https://cs.nyu.edu/~fergus/papers/fwt_ssl.pdf`

Online Decision Making on Graphs



Online Decision Making on Graphs: Smoothness

- Sequential decision making in structured settings
  - we are asked to pick a node (or a few nodes) in a graph
  - the graph encodes some **structural property** of the setting
  - goal: maximize the sum of the outcomes
  - application: recommender systems
- Exploiting **smoothness**
  - fixed graph
  - iid outcomes
  - neighboring nodes have similar outcomes

Online Decision Making on Graphs

Movie recommendation: (in each time step)

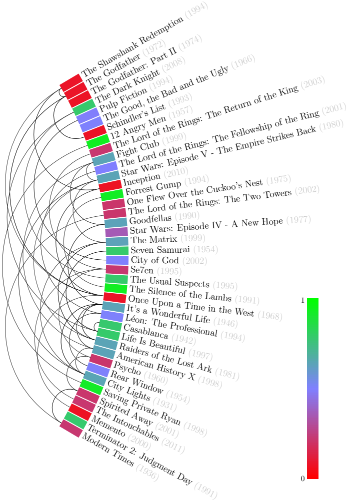
- Recommend movies to a **single user**.
- Good prediction after a few steps ($T \ll N$).

Goal:

- Maximize overall reward (sum of ratings).

Assumptions:

- Unknown reward function $f : V(G) \to \mathbb{R}$.
- Function f is **smooth** on a graph.
- Neighboring movies $\Rightarrow$ similar preferences.
- Similar preferences $\not\Rightarrow$ neighboring movies.



Recap: Smooth graph functions

- $\mathbf{f} = (f_1, \ldots, f_N)^\mathsf{T}$: Vector of function values.
- Let $\mathcal{L} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\mathsf{T}$ be the eigendecomposition of the Laplacian.
  - Diagonal matrix $\mathbf{\Lambda}$ whose diagonal entries are eigenvalues of $\mathcal{L}$.
  - Columns of $\mathbf{Q}$ are eigenvectors of $\mathcal{L}$.
  - Columns of $\mathbf{Q}$ form a basis.
- $\boldsymbol{\alpha}^*$: Unique vector such that $\mathbf{Q}\boldsymbol{\alpha}^* = \mathbf{f}$ Note: $\mathbf{Q}^\mathsf{T}\mathbf{f} = \boldsymbol{\alpha}^*$

$$S_G(\mathbf{f}) = \mathbf{f}^\mathsf{T}\mathcal{L}\mathbf{f} = \mathbf{f}^\mathsf{T}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\mathsf{T}\mathbf{f} = \boldsymbol{\alpha}^{*\mathsf{T}}\mathbf{\Lambda}\boldsymbol{\alpha}^* = \|\boldsymbol{\alpha}^*\|_{\mathbf{\Lambda}}^2 = \sum_{i=1}^N \lambda_i (\alpha_i^*)^2$$

Smoothness and <u>regularization</u>: Small value of

(a) $S_G(\mathbf{f})$ **(b)** $\mathbf{\Lambda}$ norm of $\boldsymbol{\alpha}^*$ **(c)** α_i^* for large λ_i

Online Learning Setting - Bandit Problem

Learning setting for a bandit algorithm π

- In each time t step choose a node $\pi(t)$.
- the $\pi(t)$-th row $\mathbf{x}_{\pi(t)}$ of the matrix $\mathbf{Q}$ corresponds to the arm $\pi(t)$.
- Obtain noisy reward $r_t = \mathbf{x}_{\pi(t)}^\mathsf{T}\boldsymbol{\alpha}^* + \varepsilon_t$. Note: $\mathbf{x}_{\pi(t)}^\mathsf{T}\boldsymbol{\alpha}^* = f_{\pi(t)}$
  - ε_t is R-sub-Gaussian noise. $\forall \xi \in \mathbb{R}, \mathbb{E}[e^{\xi\varepsilon_t}] \leq \exp\left(\xi^2 R^2/2\right)$

- Minimize cumulative regret

$$R_T = T \max_a \left(\mathbf{x}_a^\mathsf{T}\boldsymbol{\alpha}^*\right) - \sum_{t=1}^{T} \mathbf{x}_{\pi(t)}^\mathsf{T}\boldsymbol{\alpha}^*.$$

Can't we just use *linear bandits*?

Online Decision Making on Graphs: Smoothness

- **Linear bandit algorithms**
  - **LinUCB** (Li et al., 2010)
    - Regret bound $\approx D\sqrt{T \ln T}$
  - **LinearTS** (Agrawal and Goyal, 2013)
    - Regret bound $\approx D\sqrt{T \ln N}$

Note: D is ambient dimension, in our case N, length of x_i.
Number of actions, e.g., all possible movies $\rightarrow$ **HUGE!**

- **Spectral bandit algorithms**
  - **SpectralUCB**
    - Regret bound $\approx d\sqrt{T \ln T}$
    - Operations per step: $D^2 N$
  - **SpectralTS**
    - Regret bound $\approx d\sqrt{T \ln N}$
    - Operations per step: $D^2 + DN$

Note: d is effective dimension, usually much smaller than D.

Effective dimension

- **Effective dimension:** Largest d such that

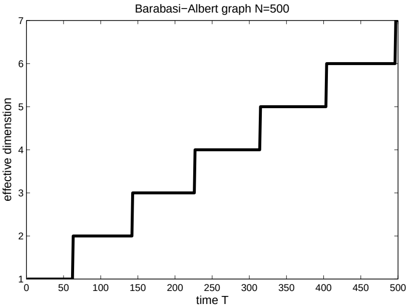
$$(d-1)\lambda_d \leq \frac{T}{\log(1 + T/\lambda)}.$$

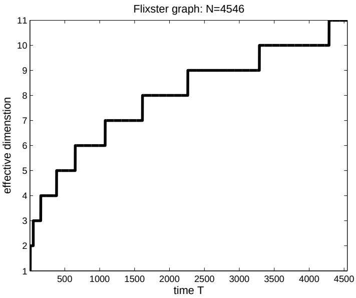
- Function of time horizon and graph properties
- λ_i : i-th smallest eigenvalue of $\mathbf{\Lambda}$.
- λ: Regularization parameter of the algorithm.

Properties:

- d is small when the coefficients λ_i grow rapidly above time.
- d is related to the number of "non-negligible" dimensions.
- Usually d is much smaller than D in real world graphs.
- Can be computed beforehand.

Effective dimension vs. Ambient dimension

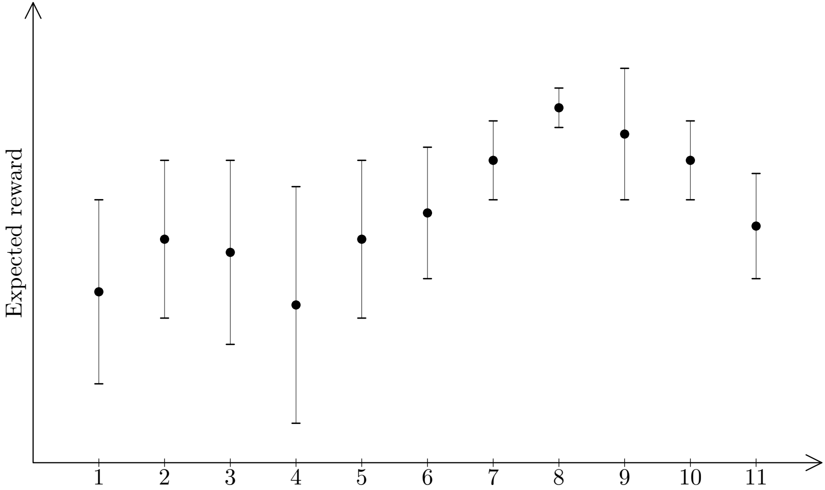




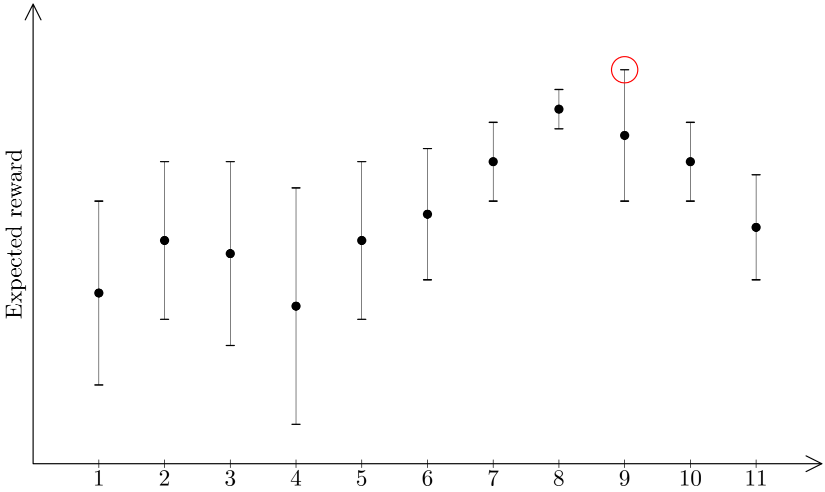
$$d \ll D$$

Note: In our setting $T < N = D$.

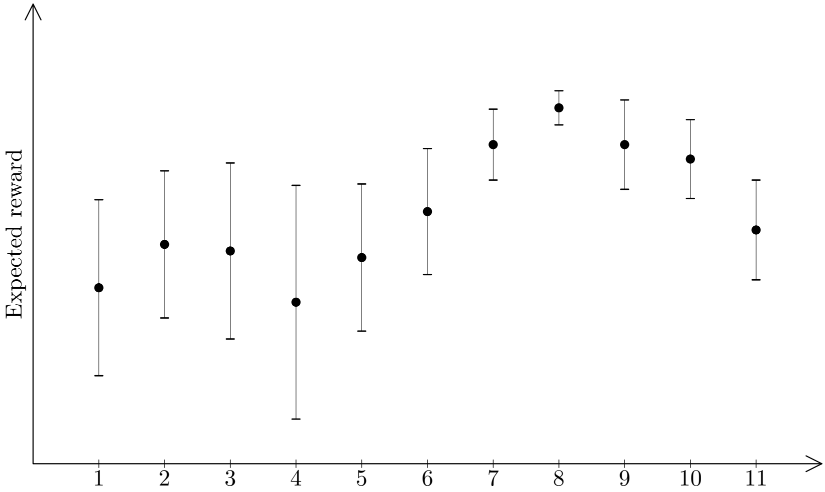
UCB style algorithms: Estimate



UCB style algorithms: Sample



UCB style algorithms: Estimate ...



SpectralUCB

Input:

N, T, $\{\mathbf{\Lambda}_{\mathcal{L}}, \mathbf{Q}\}$, λ, δ, R, C $\mathcal{L}$

Run:

$\mathbf{\Lambda} \leftarrow \mathbf{\Lambda}_{\mathcal{L}} + \lambda \mathbf{I}$

$d \leftarrow \max\{d : (d-1)\lambda_d \leq T / \ln(1 + T/\lambda)\}$

for $t = 1$ **to** T **do**

Update the basis coefficients $\hat{\alpha}$:

$\mathbf{X}_t \leftarrow [\mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(t-1)}]^{\mathsf{T}}$

$\mathbf{r} \leftarrow [r_1, \ldots, r_{t-1}]^{\mathsf{T}}$

$\mathbf{V}_t \leftarrow \mathbf{X}_t \mathbf{X}_t^{\mathsf{T}} + \mathbf{\Lambda}$

$\hat{\alpha}_t \leftarrow \mathbf{V}_t^{-1} \mathbf{X}_t^{\mathsf{T}} \mathbf{r}$

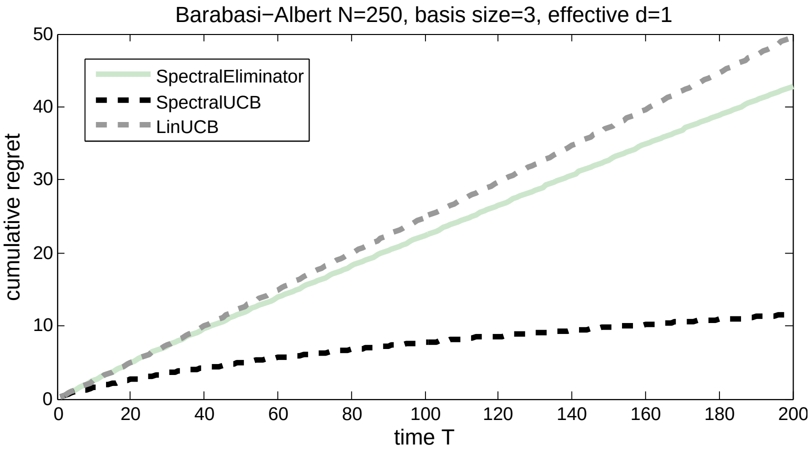
$c_t \leftarrow 2R\sqrt{d \ln(1 + t/\lambda) + 2\ln(1/\delta)} + C$

$\pi(t) \leftarrow \arg\max_a \left(\mathbf{x}_a^{\mathsf{T}} \hat{\alpha} + c_t \|\mathbf{x}_a\|_{\mathbf{V}_t^{-1}}\right)$

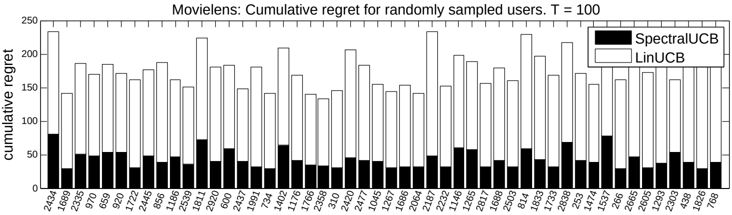
Observe the reward r_t

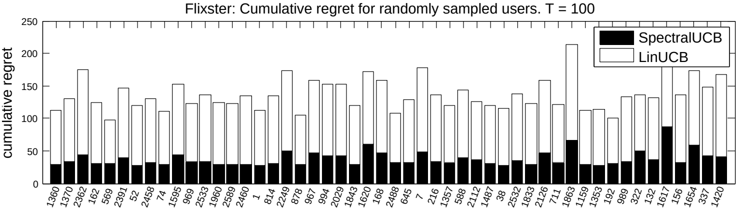
end for

SpectralUCB: Synthetic experiment



SpectralUCB: Real world experiment





SpectralUCB: Regret Bound

- d: Effective dimension.
- λ: Minimal eigenvalue of $\mathbf{\Lambda} = \mathbf{\Lambda}_{\mathcal{L}} + \lambda \mathbf{I}$.
- C: Smoothness upper bound, $\|\boldsymbol{\alpha}^*\|_{\mathbf{\Lambda}} \leq C$.
- $\mathbf{x}_i^\top \boldsymbol{\alpha}^* \in [-1, 1]$ for all i.

The **cumulative regret** R_T of **SpectralUCB** is with probability $1 - \delta$ bounded as

$$R_T \leq \left(8R\sqrt{d \ln \frac{\lambda + T}{\lambda} + 2 \ln \frac{1}{\delta}} + 4C + 4 \right) \sqrt{dT \ln \frac{\lambda + T}{\lambda}}.$$

$$R_T \approx d\sqrt{T \ln T}$$

SpectralUCB: Regret Bound

- Derivation of the confidence ellipsoid for $\hat{\alpha}$ with probability $1-\delta$.
  - Using analysis of OFUL (Abbasi-Yadkori et al., 2011)

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

- Regret in one time step: $r_t = \mathbf{x}_*^{\mathsf{T}}\alpha^* - \mathbf{x}_{\pi(t)}^{\mathsf{T}}\alpha^* \leq 2c_t\|\mathbf{x}_{\pi(t)}\|_{\mathbf{V}_t^{-1}}$
- Cumulative regret:

$$R_T = \sum_{t=1}^{T} r_t \leq \sqrt{T\sum_{t=1}^{T} r_t^2} \leq 2(c_T + 1)\sqrt{2T\ln\frac{|\mathbf{V}_T|}{|\mathbf{\Lambda}|}}$$

- Upperbound for $\ln(|\mathbf{V}_t|/|\mathbf{\Lambda}|)$

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

SpectralUCB: Regret Bound

Sylvester's determinant theorem:

$$|\mathbf{A} + \mathbf{x}\mathbf{x}^\mathsf{T}| = |\mathbf{A}||\mathbf{I} + \mathbf{A}^{-1}\mathbf{x}\mathbf{x}^\mathsf{T}| = |\mathbf{A}|(1 + \mathbf{x}^\mathsf{T}\mathbf{A}^{-1}\mathbf{x})$$

Goal:

- Upperbound determinant $|\mathbf{A} + \mathbf{x}\mathbf{x}^\mathsf{T}|$ for $\|\mathbf{x}\|_2 \leq 1$
- Upperbound $\mathbf{x}^\mathsf{T}\mathbf{A}^{-1}\mathbf{x}$

$$\mathbf{x}^\mathsf{T}\mathbf{A}^{-1}\mathbf{x} = \mathbf{x}^\mathsf{T}\mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^\mathsf{T}\mathbf{x} = \mathbf{y}^\mathsf{T}\mathbf{\Lambda}^{-1}\mathbf{y} = \sum_{i=1}^{N} \lambda_i^{-1} y_i^2$$

- $\|\mathbf{y}\|_2 \leq 1$.
- $\mathbf{y}$ is a canonical vector.
- $\mathbf{x} = \mathbf{Q}\mathbf{y}$ is an eigenvector of $\mathbf{A}$.

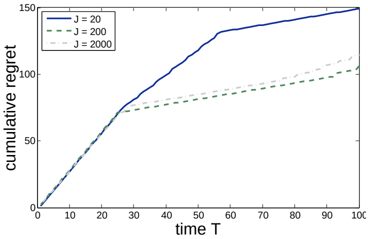
SpectralUCB: Regret Bound

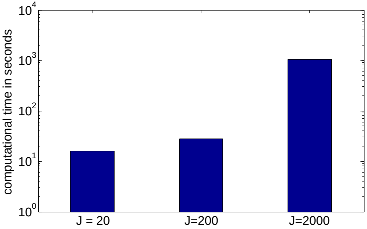
Corollary: Determinant $|\mathbf{V}_T|$ of $\mathbf{V}_T = \mathbf{\Lambda} + \sum_{t=1}^{T} \mathbf{x}_t \mathbf{x}_t^\mathsf{T}$ is maximized when all $\mathbf{x}_t$ are aligned with axes.

$$
\begin{aligned}
|\mathbf{V}_T| &\leq \max_{\sum t_i = T} \prod (\lambda_i + t_i) \\
\ln \frac{|\mathbf{V}_T|}{|\mathbf{\Lambda}|} &\leq \max_{\sum t_i = T} \sum \ln\left(1 + \frac{t_i}{\lambda_i}\right) \\
\ln \frac{|\mathbf{V}_T|}{|\mathbf{\Lambda}|} &\leq \sum_{i=1}^{d} \ln\left(1 + \frac{T}{\lambda}\right) + \sum_{i=d+1}^{N} \ln\left(1 + \frac{t_i}{\lambda_{d+1}}\right) \\
&\leq d \ln\left(1 + \frac{T}{\lambda}\right) + \frac{T}{\lambda_{d+1}} \\
&\leq 2d \ln\left(1 + \frac{T}{\lambda}\right)
\end{aligned}
$$

SpectralUCB: Improving the running time

- **Reduced basis:** We only need first few eigenvectors.
- **Getting J eigenvectors:** $\mathcal{O}(Jm \log m)$ time for m edges
- Computationally less expensive, comparable performance.

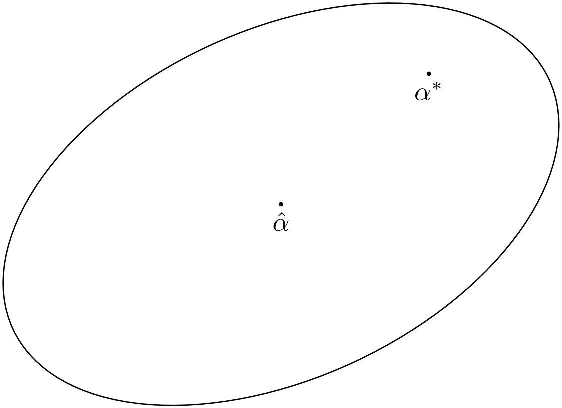




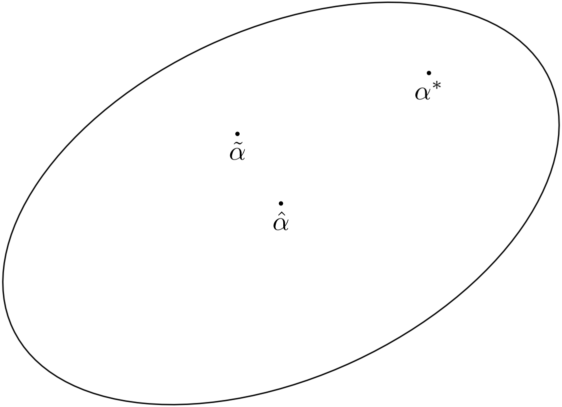
SpectralUCB: How to make it even faster?

- UCB-style algorithms need to (re)-compute UCBs every t
- Can be a problem for large set of arms $\rightarrow D^2 N \rightarrow N^3$
- Optimistic (UCB) approach vs. Thompson Sampling
  - Play the arm maximizing probability of being the best
    - Sample $\tilde{\boldsymbol{\mu}}$ from the distribution $\mathcal{N}(\hat{\boldsymbol{\mu}}, v^2\mathbf{B}^{-1})$
    - Play arm which maximizes $\mathbf{b}^\top \tilde{\boldsymbol{\mu}}$ and observe reward
  - Compute posterior distribution according to reward received
- Only requires $D^2 + DN \rightarrow N^2$ per step update

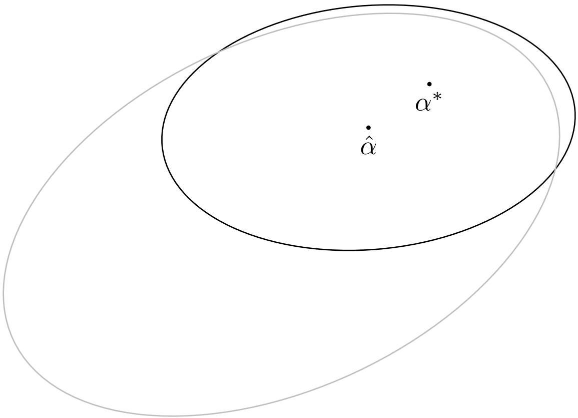
Thomson Sampling: Estimate



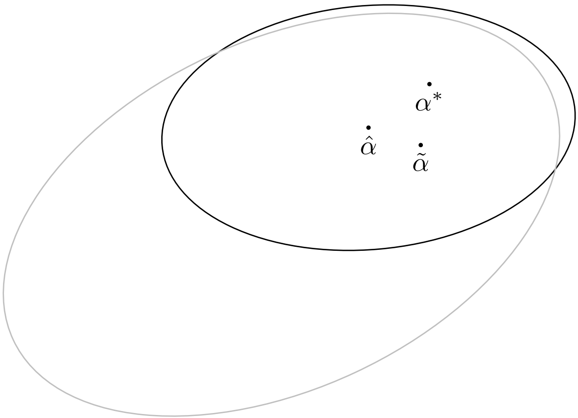
Thomson Sampling: Sample



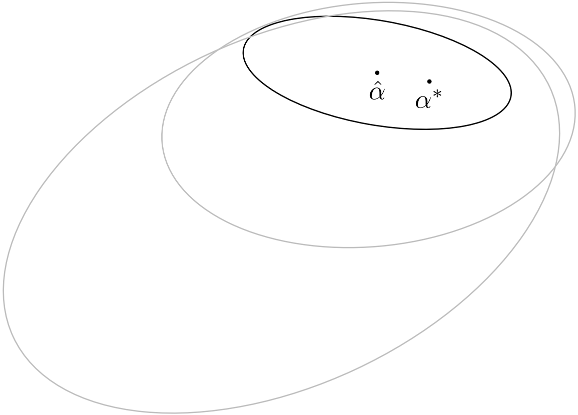
Thomson Sampling: Estimate



Thomson Sampling: Sample



Thomson Sampling: Estimate ...



SpectralTS for Graphs

Input:
N, T, $\{\mathbf{\Lambda}_{\mathcal{L}}, \mathbf{Q}\}$, λ, δ, R, C
Initialization:
$v = R\sqrt{6d\log((\lambda + T)/\delta\lambda)} + C$
$\hat{\boldsymbol{\alpha}} = 0_N$
$\mathbf{f} = 0_N$
$\mathbf{V} = \mathbf{\Lambda}_{\mathcal{L}} + \lambda \mathbf{I}_N$
Run:
for $t = 1$ **to** T **do**
Sample $\tilde{\boldsymbol{\alpha}} \sim \mathcal{N}(\hat{\boldsymbol{\alpha}}, v^2\mathbf{V}^{-1})$
$\pi(t) \leftarrow \arg\max_a \mathbf{x}_a^{\mathsf{T}}\tilde{\boldsymbol{\alpha}}$
Observe a noisy reward $r(t) = \mathbf{x}_{\pi(t)}^{\mathsf{T}}\boldsymbol{\alpha}^* + \varepsilon_t$
$\mathbf{f} \leftarrow \mathbf{f} + \mathbf{x}_{\pi(t)} r(t)$
Update $\mathbf{V} \leftarrow \mathbf{V} + \mathbf{x}_{\pi(t)}\mathbf{x}_{\pi(t)}^{\mathsf{T}}$
Update $\hat{\boldsymbol{\alpha}} \leftarrow \mathbf{V}^{-1}\mathbf{f}$
end for

SpectralTS: Regret bound

- d: Effective dimension.
- λ: Minimal eigenvalue of $\mathbf{\Lambda} = \mathbf{\Lambda}_{\mathcal{L}} + \lambda \mathbf{I}$.
- C: Smoothness upper bound, $\|\boldsymbol{\alpha}^*\|_{\mathbf{\Lambda}} \leq C$.
- $\mathbf{x}_i^{\mathsf{T}} \boldsymbol{\alpha}^* \in [-1, 1]$ for all i.

The **cumulative regret** R_T of **SpectralTS** is with probability $1 - \delta$ bounded as

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

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

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

$$R_T \approx d \sqrt{T \log N}$$

SpectralTS: Analysis sketch

Divide arms into two groups

- $\Delta_i = \mathbf{b}_*^\mathsf{T}\boldsymbol{\mu} - \mathbf{b}_i^\mathsf{T}\boldsymbol{\mu} \leq g\|\mathbf{b}_i\|_{\mathbf{B}_t^{-1}}$ arm i is **unsaturated**
- $\Delta_i = \mathbf{b}_*^\mathsf{T}\boldsymbol{\mu} - \mathbf{b}_i^\mathsf{T}\boldsymbol{\mu} > g\|\mathbf{b}_i\|_{\mathbf{B}_t^{-1}}$ arm i is **saturated**

Saturated arm

- Small standard deviation $\rightarrow$ accurate regret estimate.
- **High regret** on playing the arm $\rightarrow$ **Low probability** of picking

Unsaturated arm

- **Low regret** bounded by a factor of standard deviation
- **High probability** of picking

SpectralTS: Analysis sketch

- Confidence ellipsoid for estimate $\hat{\boldsymbol{\mu}}$ of $\boldsymbol{\mu}$ (with probability $1 - \delta/T^2$)
  - Using analysis of OFUL algorithm (Abbasi-Yadkori et al., 2011)

$$|\mathbf{b}_i^\mathsf{T}\hat{\boldsymbol{\mu}} - \mathbf{b}_i^\mathsf{T}\boldsymbol{\mu}| \leq \left(R\sqrt{2\,d\log\left(\frac{(\lambda+T)T^2}{\delta\lambda}\right)} + C\right)\|\mathbf{b}_i\|_{\mathbf{B}_t^{-1}} = \ell\|\mathbf{b}_i\|_{\mathbf{B}_t^{-1}}$$

- The key result coming from spectral properties of $\mathbf{B}_t$.

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

- Concentration of sample $\tilde{\boldsymbol{\mu}}$ around mean $\hat{\boldsymbol{\mu}}$ (with probability $1 - 1/T^2$)
  - Using concentration inequality for Gaussian random variable.

$$|\mathbf{b}_i^\mathsf{T}\tilde{\boldsymbol{\mu}} - \mathbf{b}_i^\mathsf{T}\hat{\boldsymbol{\mu}}| \leq \left(R\sqrt{6d\log\left(\frac{\lambda+T}{\delta\lambda}\right)} + C\right)\|\mathbf{b}_i\|_{\mathbf{B}_t^{-1}}\sqrt{4\log(TN)} = v\|\mathbf{b}_i\|_{\mathbf{B}_t^{-1}}\sqrt{4\log(TN)}$$

SpectralTS: Analysis sketch

Define $\text{regret}'(t) = \text{regret}(t) \cdot \mathbb{1}\{|\mathbf{b}_i^\mathsf{T}\hat{\boldsymbol{\mu}}(t) - \mathbf{b}_i^\mathsf{T}\boldsymbol{\mu}| \leq \ell \|\mathbf{b}_i\|_{\mathbf{B}_t^{-1}}\}$

$$\text{regret}'(t) \leq \frac{11g}{p}\|\mathbf{b}_{a(t)}\|_{\mathbf{B}_t^{-1}} + \frac{1}{T^2}$$

Super-martingale (i.e. $\mathbb{E}[Y_t - Y_{t-1}|\mathcal{F}_{t-1}] \leq 0$)

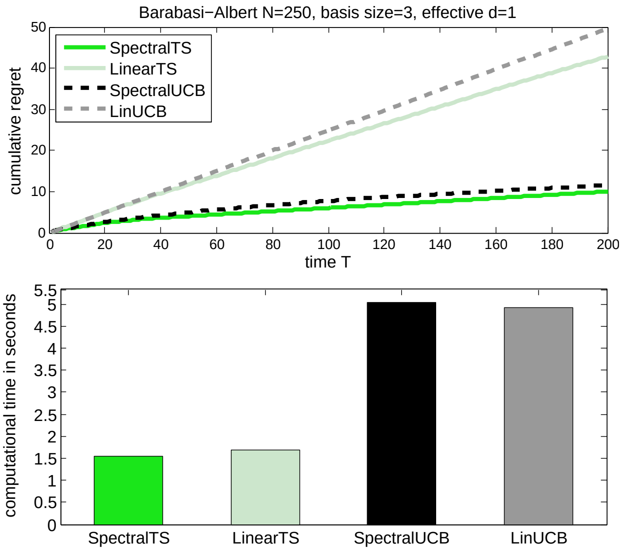
$$X_t = \text{regret}'(t) - \frac{11g}{p}\|\mathbf{b}_{a(t)}\|_{\mathbf{B}_t^{-1}} - \frac{1}{T^2}$$
$$Y_t = \sum_{w=1}^{t} X_w.$$

$(Y_t;\ t = 0, \ldots, T)$ is a **super-martingale** process w.r.t. history $\mathcal{F}_t$.

Azuma-Hoeffding inequality for super-martingale, w. p. $1 - \delta/2$:

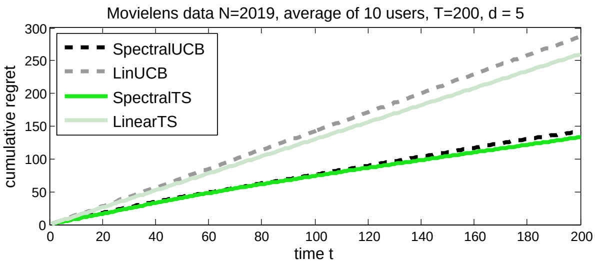
$$\sum_{t=1}^{T} \text{regret}'(t) \leq \frac{11g}{p}\sum_{t=1}^{T}\|\mathbf{b}_{a(t)}\|_{\mathbf{B}_t^{-1}} + \frac{1}{T} + \frac{g}{p}\left(\frac{11}{\sqrt{\lambda}} + 2\right)\sqrt{2T\ln\frac{2}{\delta}}$$

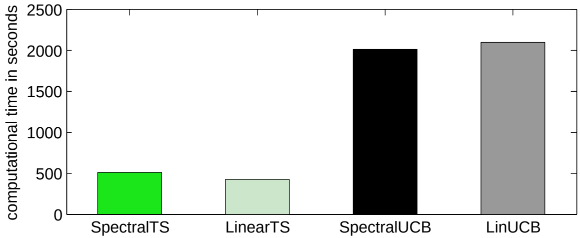
Spectral Bandits: Synthetic experiment



Spectral Bandits: Real world experiment

MovieLens dataset of 6k users who rated one million movies.





Spectral Bandits Summary

- Spectral bandit setting **(smooth graph functions)**.
- **SpectralUCB**
  - Regret bound $\approx d\sqrt{T \ln T}$
- **SpectralTS**
  - Regret bound $\approx d\sqrt{T \ln N}$
  - Computationally more efficient.
- **SpectralEliminator**
  - Regret bound $\approx \sqrt{dT \ln T}$
  - Better upper, empirically does not seem to work well (yet)
- Bounds scale with **effective dimension** $d \ll D$.

SpectralEliminator: Pseudocode

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:

$\mathbf{x}_t \leftarrow \arg\max_{\mathbf{x} \in A_j} \|\mathbf{x}\|_{\mathbf{V}_t^{-1}}$

$\mathbf{V}_{t+1} \leftarrow \mathbf{V}_t + \mathbf{x}_t \mathbf{x}_t^\mathsf{T}$

end for

Eliminate the arms that are not promising:

$\hat{\alpha}_t \leftarrow \mathbf{V}_t^{-1} [\mathbf{x}_{t_j}, \ldots, \mathbf{x}_t][r_{t_j}, \ldots, r_t]^\mathsf{T}$

$A_{j+1} \leftarrow \left\{ \mathbf{x} \in A_j, \langle \hat{\alpha}_t, \mathbf{x} \rangle + \|\mathbf{x}\|_{V_t^{-1}} \beta \geq \max_{\mathbf{x} \in A_j} \left[\langle \hat{\alpha}_t, \mathbf{x} \rangle - \|\mathbf{x}\|_{\mathbf{V}_t^{-1}} \beta \right] \right\}$

end for

SpectralEliminator: Analysis

SpectralEliminator

- Divide time into sets $(t_1 = 1 \leq t_2 \leq \dots)$ to introduce independence for Azuma-Hoeffding inequality and observe $R_T \leq \sum_{j=0}^{J}(t_{j+1} - t_j)\left[\langle \mathbf{x}^* - \mathbf{x}_t, \hat{\alpha}_j \rangle + (\|\mathbf{x}^*\|_{\mathbf{V}_j^{-1}} + \|\mathbf{x}_t\|_{\mathbf{V}_j^{-1}})\beta\right]$
- Bound $\langle \mathbf{x}^* - \mathbf{x}_t, \hat{\alpha}_j \rangle$ for each phase
- No bad arms: $\langle \mathbf{x}^* - \mathbf{x}_t, \hat{\alpha}_j \rangle \leq (\|\mathbf{x}^*\|_{\mathbf{V}_j^{-1}} + \|\mathbf{x}_t\|_{\mathbf{V}_j^{-1}})\beta$
- By algorithm: $\|\mathbf{x}\|^2_{\mathbf{V}_j^{-1}} \leq \frac{1}{t_j - t_{j-1}} \sum_{s=t_{j-1}+1}^{t_j} \|\mathbf{x}_s\|^2_{\mathbf{V}_{s-1}^{-1}}$
- $\sum_{s=t_{j-1}+1}^{t_j} \min\left(1, \|\mathbf{x}_s\|^2_{\mathbf{V}_{s-1}^{-1}}\right) \leq \log \frac{|\mathbf{V}_j|}{|\mathbf{\Lambda}|}$

Michal Valko
michal.valko@inria.fr
sequel.lille.inria.fr