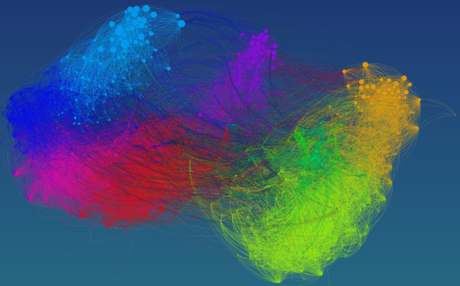
Graphs in Machine Learning

Michal Valko

Inria Lille - Nord Europe, France

Partially based on material by: Tomáš Kocák, Nikhil Srivastava, Yiannis Koutis, Joshua Batson, Daniel Spielman



Last 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

This Lecture

- Graph bandits and online non-stochastic rewards
- Observability graphs
- Side information
- Influence Maximization
- Graph Sparsification
- Spectral Sparsification

Previous Lab Session

- 16. 11. 2015 by Daniele.Calandriello@inria.fr
- Content
  - Semi-supervised learning
  - Graph quantization
  - Online face recognizer
- Short written report
- Questions to piazza
- *Deadline:* 30. 11. 2015 (today)
- `http://researchers.lille.inria.fr/~calandri/teaching.html`

Next Lab Session

- 7. 12. 2015 by Daniele.Calandriello@inria.fr
- Content
  - GraphLab
  - Large-Scale Graph Learning
- AR: Get the GraphLab license
- AR: Refresh Python
  - `http://learnxinyminutes.com/docs/python/` *strongly recommended*
  - `https://www.codecademy.com/tracks/python` *crash course*
- Short written report
- Questions to piazza
- *Deadline:* 21. 12. 2015
- `http://researchers.lille.inria.fr/~calandri/teaching.html`

Final class projects

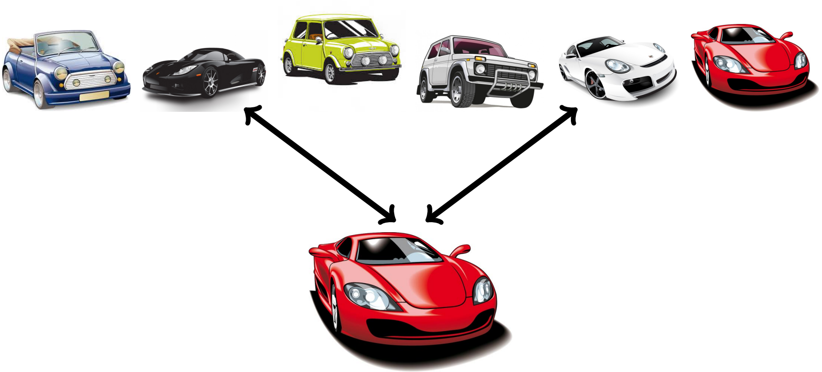
- time and formatting description on the class website
- grade: report + short presentation of the **team**
- deadlines
  - 30. 11. 2015 (today)
  - 6. 1. 2016 final report (for all projects)
  - 20. 1. 2016, presentation in class from 14h00 (Cournot C102)
  - alternatively Jan 2016, remote presentations (other projects)
- project report: 5 - 10 pages in NIPS format
- presentation: 20 minutes (**time it!**), everybody has to present
- book presentation time slot on the website
- **explicitly state the contributions**

Online Decision Making on Graphs

- 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
- Specific applications
  - *First application:* **smoothness**
  - *Second application:* **side information**
  - *Third application:* **influence maximization**

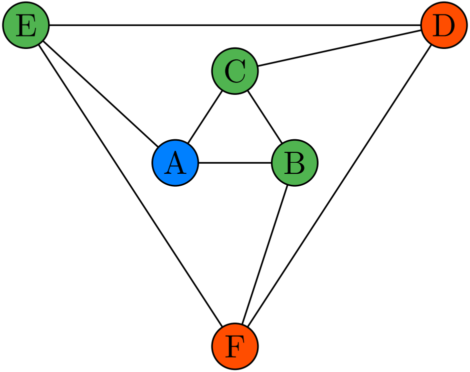
Graph bandits: Side observations

Example 1: undirected observations



Graph bandits: Side observations

Example 1: Graph Representation

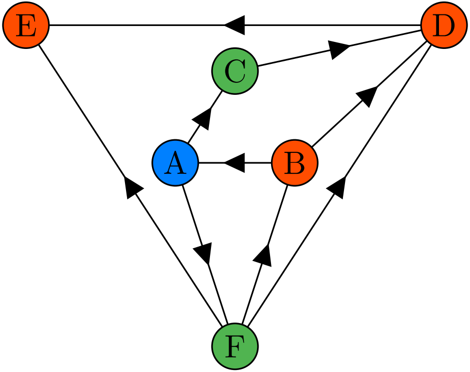


Graph bandits: Side observations

Example 2: Directed observation

Graph bandits: Side observations

Example 2



Graph bandits: Side observations

Learning setting

In each time step $t = 1, \ldots, T$

- **Environment (adversary):**
  - Privately assigns losses to actions
  - Generates an observation graph
    - Undirected / Directed
    - Disclosed / Not disclosed
- **Learner:**
  - Plays action $I_t \in [N]$
  - Obtain loss ℓ_{t,I_t} of action played
  - Observe losses of neighbors of I_t
    - Graph: disclosed
- **Performance measure:** Total expected regret

$$R_T = \max_{i \in [N]} \mathbb{E}\left[\sum_{t=1}^{T} (\ell_{t,I_t} - \ell_{t,i})\right]$$

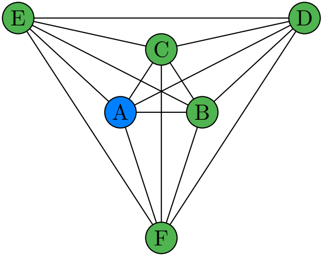
Graph bandits: Typical settings

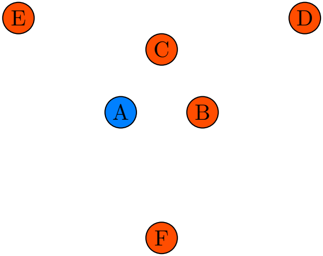
Full Information setting

- Pick an action (e.g. action A)
- Observe losses of all actions
- $R_T = \widetilde{\mathcal{O}}(\sqrt{T})$

Bandit setting

- Pick an action (e.g. action A)
- Observe loss of a chosen action
- $R_T = \widetilde{\mathcal{O}}(\sqrt{NT})$

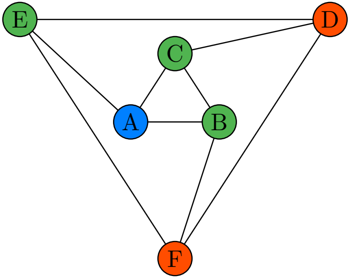




Graph bandits: Side observation - Undirected case

Side observation (Undirected case)

- Pick an action (e.g. action A)
- Observe losses of neighbors

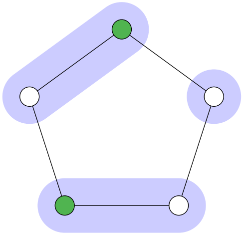


Mannor and Shamir (ELP algorithm)

- Need to know the graph
- Clique decomposition (c cliques)
- $R_T = \widetilde{\mathcal{O}}(\sqrt{cT})$

Alon, Cesa-Bianchi, Gentile, Mansour

- No need to know the graph
- Independence set of α actions
- $R_T = \widetilde{\mathcal{O}}(\sqrt{\alpha T})$



Graph bandits: Side observation - Directed case

Side observation (Directed case)

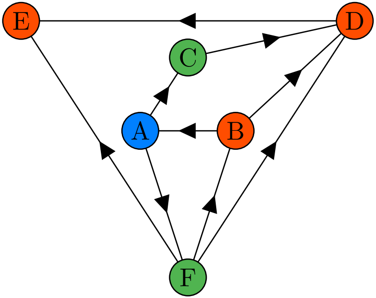
- Pick an action (e.g. action A)
- Observe losses of neighbors

Alon, Cesa-Bianchi, Gentile, Mansour

- **Exp3-DOM**
- Need to know graph
- Need to find dominating set
- $R_T = \widetilde{\mathcal{O}}(\sqrt{\alpha T})$

Exp3-IX - Kocák et. al

- No need to know graph
- $R_T = \widetilde{\mathcal{O}}(\sqrt{\alpha T})$



Reminder: Exp3 algorithms in general

- **Compute weights** using loss estimates $\hat{\ell}_{t,i}$.

$$w_{t,i} = \exp\left(-\eta \sum_{s=1}^{t-1} \hat{\ell}_{s,i}\right)$$

- **Play action** I_t such that

$$\mathbb{P}(I_t = i) = p_{t,i} = \frac{w_{t,i}}{W_t} = \frac{w_{t,i}}{\sum_{j=1}^{N} w_{t,j}}$$

- **Update loss estimates** (using observability graph)

How the algorithms approach the bias-variance tradeoff?

Bias variance tradeoff approaches

- Approach of Mixing
  - Bias sampling distribution $\mathbf{p}_t$ over actions
    - $\mathbf{p}'_t = (1-\gamma)\mathbf{p}_t + \gamma\mathbf{s}_t$ – mixed distribution
    - $\mathbf{s}_t$ – probability distribution which supports exploration
  - Loss estimates $\hat{\ell}_{t,i}$ are unbiased

- Approach of Implicit eXploration (**IX**)
  - Bias loss estimates $\hat{\ell}_{t,i}$
    - Biased loss estimates $\implies$ biased weights
    - Biased weights $\implies$ biased probability distribution
  - No need for mixing

Is there a difference in a traditional non-graph case? Not much

Big difference in graph feedback case!

Graph bandits: Mannor and Shamir - ELP algorithm

- $\mathbb{E}[\widehat{\ell}_{t,i}] = \ell_{t,i}$ – unbiased loss estimates
- $p'_{t,i} = (1-\gamma)p_{t,i} + \gamma s_{t,i}$ – bias by mixing
- $\mathbf{s}_t = \{s_{t,1}, \ldots, s_{t,N}\}$ – probability distribution over the action set

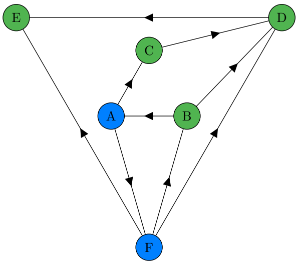
$$\mathbf{s}_t = \arg\max_{\mathbf{s}_t} \left[\min_{j\in[N]} \left(s_{t,j} + \sum_{k\in N_{t,j}} s_{t,k} \right) \right] = \arg\max_{\mathbf{s}_t} \left[\min_{j\in[N]} q_{t,j} \right]$$

- $q_{t,j}$ – probability that loss of j is observed according to $\mathbf{s}_t$
- **Computation of $\mathbf{s}_t$**
  - Graph needs to be disclosed
  - Solving simple linear program
- Needs to know graph before playing an action
- Graphs can be only undirected

Graph bandits: Alon, Cesa-Bianchi, Gentile, Mansour - Exp3-DOM

- $\mathbb{E}[\widehat{\ell}_{t,i}] = \ell_{t,i}$ – unbiased loss estimates
- $p'_{t,i} = (1-\gamma)p_{t,i} + \gamma s_{t,i}$ – bias by mixing
- $\mathbf{s}_t = \{s_{t,1}, \ldots, s_{t,N}\}$ – probability distribution over the action set

$$s_{t,i} = \begin{cases} \frac{1}{r} & \text{if } i \in R;\ |R| = r \\ 0 & \text{otherwise.} \end{cases}$$



- R – dominating set of r elements
- $\mathbf{s}_t$ – uniform distribution over R
- Needs to know graph beforehand
- Graphs can be directed

Graph bandits: Comparison of loss estimates

Typical algorithms - loss estimates

$$\hat{\ell}_{t,i} = \begin{cases} \ell_{t,i}/o_{t,i} & \text{if } \ell_{t,i} \text{ is observed} \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbb{E}[\hat{\ell}_{t,i}] = \frac{\ell_{t,i}}{o_{t,i}} o_{t,i} + 0(1 - o_{t,i}) = \ell_{t,i}$$

Exp3-IX - loss estimates

$$\hat{\ell}_{t,i} = \begin{cases} \ell_{t,i}/(o_{t,i} + \gamma) & \text{if } \ell_{t,i} \text{ is observed} \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbb{E}[\hat{\ell}_{t,i}] = \frac{\ell_{t,i}}{o_{t,i} + \gamma} o_{t,i} + 0(1 - o_{t,i}) = \ell_{t,i} - \ell_{t,i} \frac{\gamma}{o_{t,i} + \gamma} \leq \ell_{t,i}$$

No mixing!

Analysis of Exp3 algorithms in general

- Evolution of W_{t+1}/W_t

$$\frac{1}{\eta}\log\frac{W_{t+1}}{W_t} \le \frac{1}{\eta}\log\left(1-\eta\sum_{i=1}^{N} p_{t,i}\hat{\ell}_{t,i} + \frac{\eta^2}{2}\sum_{i=1}^{N} p_{t,i}(\hat{\ell}_{t,i})^2\right),$$

$$\sum_{i=1}^{N} p_{t,i}\hat{\ell}_{t,i} \le \left[\frac{\log W_t}{\eta} - \frac{\log W_{t+1}}{\eta}\right] + \frac{\eta}{2}\sum_{i=1}^{N} p_{t,i}(\hat{\ell}_{t,i})^2$$

- Taking expectation and summing over time

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{N} p_{t,i}\hat{\ell}_{t,i}\right] - \mathbb{E}\left[\sum_{t=1}^{T}\hat{\ell}_{t,k}\right] \le \mathbb{E}\left[\frac{\log N}{\eta}\right] + \mathbb{E}\left[\frac{\eta}{2}\sum_{t=1}^{T}\sum_{i=1}^{N} p_{t,i}(\hat{\ell}_{t,i})^2\right]$$

Graph bandits: Regret bound of Exp3-IX

$$\underbrace{\mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{N} p_{t,i}\hat{\ell}_{t,i}\right]}_{A} - \underbrace{\mathbb{E}\left[\sum_{t=1}^{T}\hat{\ell}_{t,k}\right]}_{B} \leq \mathbb{E}\left[\frac{\log N}{\eta}\right] + \underbrace{\mathbb{E}\left[\frac{\eta}{2}\sum_{t=1}^{T}\sum_{i=1}^{N} p_{t,i}(\hat{\ell}_{t,i})^2\right]}_{C}$$

Lower bound of A (using definition of loss estimates)

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{N} p_{t,i}\hat{\ell}_{t,i}\right] \geq \mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{N} p_{t,i}\ell_{t,i}\right] - \mathbb{E}\left[\gamma\sum_{t=1}^{T}\sum_{i=1}^{N}\frac{p_{t,i}}{o_{t,i}+\gamma}\right]$$

Lower bound of B (optimistic loss estimates: $\mathbb{E}[\hat{\ell}] < \mathbb{E}[\ell]$)

$$-\mathbb{E}\left[\sum_{t=1}^{T}\hat{\ell}_{t,k}\right] \geq -\mathbb{E}\left[\sum_{t=1}^{T}\ell_{t,k}\right]$$

Upper bound of C (using definition of loss estimates)

$$\mathbb{E}\left[\frac{\eta}{2}\sum_{t=1}^{T}\sum_{i=1}^{N} p_{t,i}(\hat{\ell}_{t,i})^2\right] \leq \mathbb{E}\left[\frac{\eta}{2}\sum_{t=1}^{T}\sum_{i=1}^{N}\frac{p_{t,i}}{o_{t,i}+\gamma}\right]$$

Graph bandits: Regret bound of Exp3-IX

Upper bound on regret Exp3-IX

$$R_T \leq \frac{\log N}{\eta} + \left(\frac{\eta}{2} + \gamma\right) \sum_{t=1}^{T} \mathbb{E}\left[\sum_{i=1}^{N} \frac{p_{t,i}}{o_{t,i} + \gamma}\right]$$

$$R_T \approx \mathcal{O}\left(\sqrt{\log N \sum_{t=1}^{T} \mathbb{E}\left[\sum_{i=1}^{N} \frac{p_{t,i}}{o_{t,i} + \gamma}\right]}\right)$$

Graph bandits: Regret bound of Exp3-IX

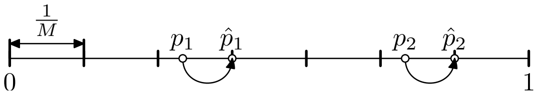
Graph lemma

- Graph G with $V(G) = \{1, \ldots, N\}$
- d_i^- – in-degree of vertex i
- α – independence set of G
- Turán's Theorem + induction

$$\sum_{i=1}^{N} \frac{1}{1 + d_i^-} \leq 2\alpha \log\left(1 + \frac{N}{\alpha}\right)$$

Graph bandits: Regret bound of Exp3-IX

Discretization



$$\sum_{i=1}^{N} \frac{p_{t,i}}{o_{t,i} + \gamma} = \sum_{i=1}^{N} \frac{p_{t,i}}{p_{t,i} + \sum_{j \in N_i^-} p_{t,j} + \gamma} \leq \sum_{i=1}^{N} \frac{\widehat{p}_{t,i}}{\widehat{p}_{t,i} + \sum_{j \in N_i^-} \widehat{p}_{t,j}} + 2$$

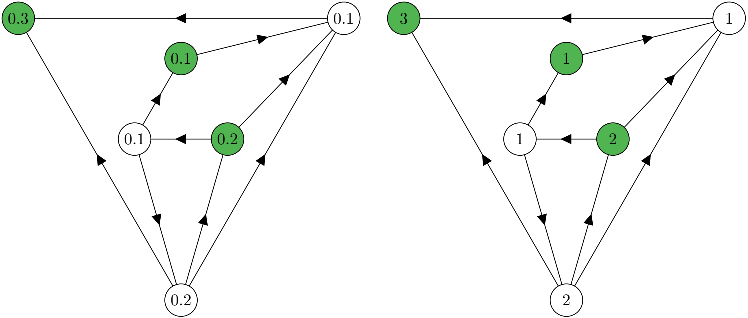
Note: we set $M = \lceil N^2/\gamma \rceil$

$$\sum_{i=1}^{N} \frac{\widehat{p}_{t,i}}{\widehat{p}_{t,i} + \sum_{j \in N_i^-} \widehat{p}_{t,j}}$$

Graph bandits: Regret bound of Exp3-IX

$$\sum_{i=1}^{N} \frac{\color{red}{M}\widehat{p}_{t,i}}{\color{red}{M}\widehat{p}_{t,i} + \sum_{j \in N_i^-} \color{red}{M}\widehat{p}_{t,j}} = \sum_{i=1}^{N} \sum_{k \in C_i} \frac{1}{1 + d_k^-} \le 2\alpha \log\left(1 + \frac{M+N}{\alpha}\right)$$

Example: let $M = 10$



Exp3-IX regret bound

$$R_T \leq \frac{\log N}{\eta} + \left(\frac{\eta}{2} + \gamma\right) \sum_{t=1}^{T} \mathbb{E}\left[2\alpha_t \log\left(1 + \frac{\lceil N^2/\gamma \rceil + N}{\alpha_t}\right) + 2\right]$$

$$R_T = \widetilde{\mathcal{O}}\left(\sqrt{\overline{\alpha}\, T \ln N}\right)$$

Next step

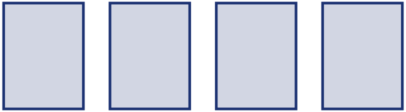
Generalization of the setting to **combinatorial actions**

Graph bandits: Complex actions

Example: Multiple Ads

Ad_{01}	Ad_{04}	Ad_{07}	Ad_{10}	Ad_{13}	Ad_{16}
Ad_{02}	Ad_{05}	Ad_{08}	Ad_{11}	Ad_{14}	Ad_{17}
Ad_{03}	Ad_{06}	Ad_{09}	Ad_{12}	Ad_{15}	Ad_{18}

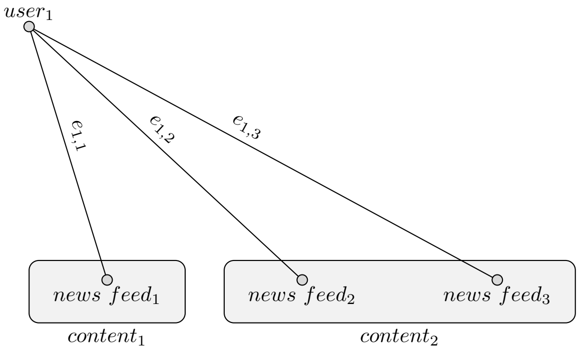
- Display 4 ads (more than 1) and observe losses

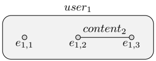


- Play m out of N actions
- Observe losses of all neighbors of played actions

Graph bandits: Complex actions

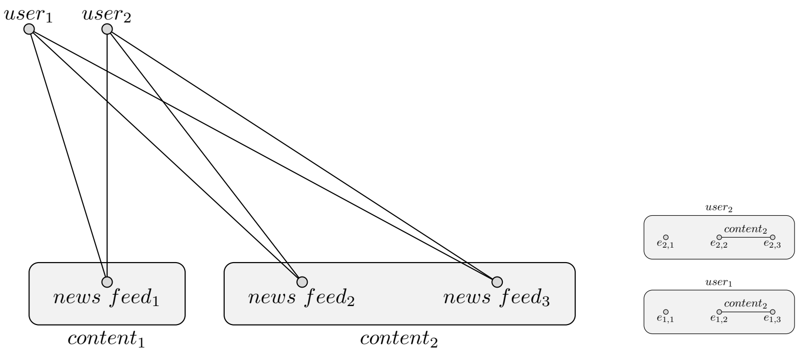
Example: New feeds





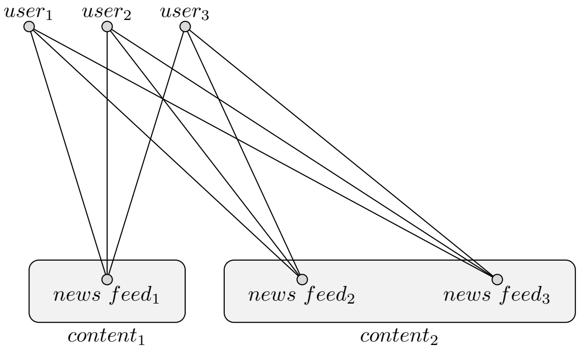
Graph bandits: Complex actions

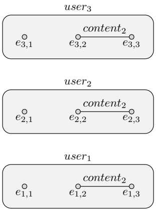
Example: New feeds



Graph bandits: Complex actions

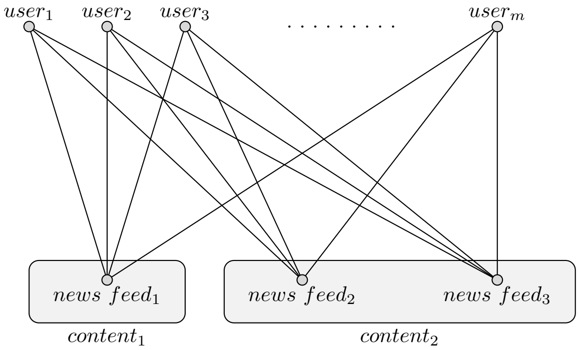
Example: New feeds

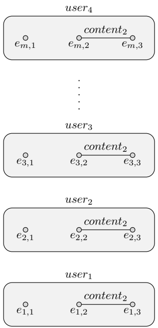




Graph bandits: Complex actions

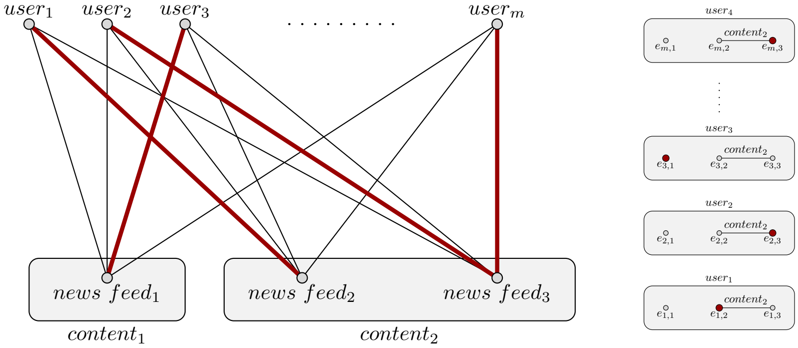
Example: New feeds





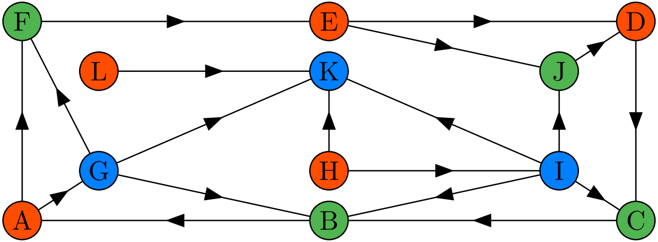
Graph bandits: Complex actions

Example: New feeds



- Play m out of N nodes (combinatorial structure)
- Obtain losses of all played nodes
- Observe losses of all neighbors of played nodes

Graph bandits: Complex actions



- Play action $\mathbf{V}_t \in S \subset \{0,1\}^N$, $\|\mathbf{v}\|_1 \leq m$ from all $\mathbf{v} \in S$
- Obtain losses $\mathbf{V}_t^\mathsf{T} \ell_t$
- Observe additional losses according to the graph

Graph bandits: FPL-IX algorithm

- **Draw perturbation** $Z_{t,i} \sim \text{Exp}(1)$ for all $i \in [N]$

- **Play "the best" action $\mathbf{V}_t$** according to total loss estimate $\widehat{\mathbf{L}}_{t-1}$ and perturbation $\mathbf{Z}_t$

$$\mathbf{V}_t = \arg\min_{\mathbf{v} \in \mathcal{S}} \mathbf{v}^\mathsf{T} \left(\eta_t \widehat{\mathbf{L}}_{t-1} - \mathbf{Z}_t \right)$$

- **Compute loss estimates**

$$\hat{\ell}_{t,i} = \ell_{t,i} K_{t,i} \mathbb{1}\{\ell_{t,i} \text{ is observed}\}$$

- $K_{t,i}$: geometric random variable with

$$\mathbb{E}[K_{t,i}] = \frac{1}{o_{t,i} + (1 - o_{t,i})\gamma}$$

Graph bandits: Complex actions

FPL-IX - regret bound

$$R_T = \widetilde{\mathcal{O}}\left(m^{3/2}\sqrt{\sum_{t=1}^{T}\alpha_t}\right) = \widetilde{\mathcal{O}}\left(m^{3/2}\sqrt{\overline{\alpha}T}\right)$$

Graph bandits: Stochastic Rewards

Can we do better if the losses/rewards are stochastic?

Yes, we can!

UCB-N - Follow UCB and update the estimates with extra info.

UCB-MaxN - Follow UCB, but pick the empirically best node in the clique of the node UCB would pick.

UCB-LP - linear approximation to the dominating set

http://www.auai.org/uai2012/papers/236.pdf

http://newslab.ece.ohio-state.edu/~buccapat/mabSigfinal.pdf

Known bounds in terms of cliques and dominating sets.

Graph bandits: Side Observation Summary

- Implicit eXploration idea
- **Algorithm for simple actions - Exp3-IX**
  - Using implicit exploration idea
  - Same regret bound as previous algorithm
  - No need to know graph before an action is played
  - Computationally efficient
- **Combinatorial setting with side observations**
- **Algorithm for combinatorial setting - FPL-IX**
- **Extensions** (open questions)
  - No need to know graph after an action is played
  - Stochastic side observations - Random graph models
  - Exploiting the communities
- **Stochastic losses**

Graph bandits: Very hot topic!

Extensions: Noga Alon et al. (2015) Beyond bandits

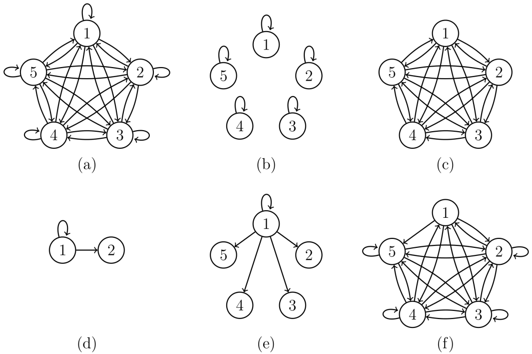


Figure 1: Examples of feedback graphs: (a) *full feedback*, (b) *bandit feedback*, (c) *loopless clique*, (d) *apple tasting*, (e) *revealing action*, (f) a clique minus a self-loop and another edge.

Complete characterization: Bártok et al. (2014)

Revealing Graph bandits: Influence Maximization

Model: Unknown $\mathbf{M} = (p_{i,j})_{i,j}$ symmetric matrix of influences

In each time step $t = 1, \ldots, T$

- learners picks a node k_t
- set $S_{k_t,t}$ of influenced nodes is *revealed*

Select influential people = Find the strategy maximizing

$$L_T = \sum_{t=1}^{T} |S_{k_t,t}| \, .$$

The number of expected influences of node k is by definition

$$r_k = \mathbb{E}\left[|S_{k,t}|\right] = \sum_{j \leq N} p_{k,j}.$$

Revealing Graph bandits: Influence Maximization

Oracle strategy always selects the best:

$$k^* = \arg\max_k \mathbb{E}\left[\sum_{t=1}^{T} |S_{k,t}|\right] = \arg\max_k Tr_k.$$

Let the reward of this node be $r_* = r_{k^*}$. Its expected performance if it consistently sampled k^* over n rounds is equal to

$$\mathbb{E}\left[L_T^*\right] = Tr^*.$$

Expected *regret* of any adaptive, non-oracle strategy unaware of **M**:

$$\mathbb{E}\left[R_T\right] = \mathbb{E}\left[L_T^*\right] - \mathbb{E}\left[L_T\right].$$

Revealing Graph bandits: Influence Maximization

Ignoring the structure again? The best we can do is $\widetilde{\mathcal{O}}\left(\sqrt{r_* T N}\right)$

We aim to do better: $R_T = \widetilde{\mathcal{O}}\left(\sqrt{r_* T D_*}\right)$

D_* - detectable dimension dependent on T and the structure

- *good case:* star-shaped graph can have $D_* = 1$
- *bad case:* a graph with many small cliques.
- *the worst case:* all nodes are disconnected except 2

Idea of the algorithm:

- *exploration phase:* sample randomly to find out $\approx D_*$ nodes
- *bandit case:* use any bandit algorithm on these nodes

More information: Revealing Graph Bandits for Maximizing Local Influence, Carpentier and Valko, AISTATS 2016

Advanced Learning for Text and Graph Data

Time: Spring term **4 lectures** and **3 Labs**
Place: Polytechnique / Amphi Sauvy
Lecturer 1: Michalis Vazirgiannis (Polytechnique)
Lecturer 2: Yassine Faihe (Hewlett-Packard - Vertica)

ALTeGraD follows after **Graphs in ML**
The two graph courses are coordinated to be complementary.

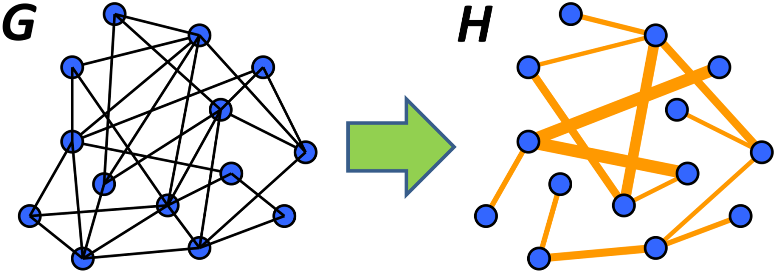
Some of covered graph topics not covered in this course

- Ranking algorithms and measures (Kendal Tau, NDCG)
- Advanced graph generators
- Community mining, advanced graph clustering
- Graph degeneracy (k-core & extensions)
- Privacy in graph mining

http://www.math.ens-cachan.fr/version-francaise/formations/master-mva/contenus-/advanced-learning-for-text-and-graph-data-altegrad--239506.kjsp?RH=1242430202531

Graph Sparsification

Goal: Get graph G and find sparse H



Why could we want to get H? smaller, faster to work with

What properties should we want from H?

Graph Sparsification: What is sparse?

What does sparse graph mean?

- average degree < 10 is pretty sparse
- for billion nodes even 100 should be ok
- in general: average degree $<$ polylog n

Are all edges important?

in a tree — sure, in a dense graph perhaps not

But real-world graphs are sparse, why care?

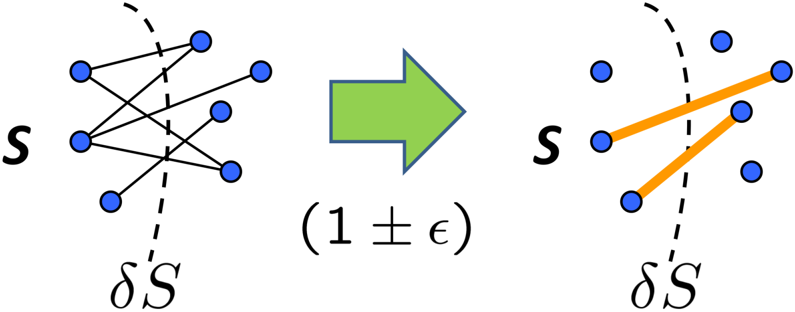
graphs that arise inside algorithms, similarity graphs, . . .

Alternative to sparsification?

example: local computation . . .

Graph Sparsification: What is good sparse?

Good sparse by Benczúr and Karger (1996) = **cut preserving**!



H approximates G well iff $\forall S \subset V$, sum of edges on δS remains

δS = edges leaving S

https://math.berkeley.edu/~nikhil/

Graph Sparsification: What is good sparse?

Good sparse by Benczúr and Karger (1996) = **cut preserving**!

Why did they care? faster mincut/maxflow

Recall what is a cut: $\mathrm{cut}_G(S) = \sum_{i \in S, j \in \overline{S}} w_{i,j}$

Define G and H are $(1 \pm \varepsilon)$-**cut similar** when $\forall S$

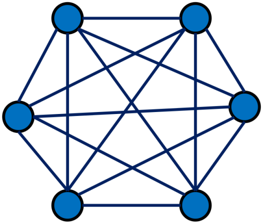
$$(1-\varepsilon)\mathrm{cut}_H(S) \leq \mathrm{cut}_G(S) \leq (1+\varepsilon)\mathrm{cut}_H(S)$$

Is this always possible? Benczúr and Karger (1996): Yes!

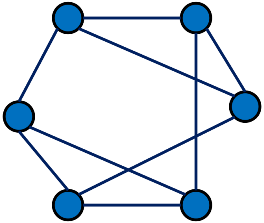
$\forall \varepsilon\ \exists\ (1+\varepsilon)$-cut similar $\widetilde{G}$ with $\mathcal{O}(n \log n / \varepsilon^2)$ edges s.t. $E_H \subseteq E$ and computable in $\mathcal{O}(m \log^3 n + m \log n / \varepsilon^2)$ time n nodes, m edges

Graph Sparsification: What is **good** sparse?

$G = K_n$



$H = d$-**regular** (random)



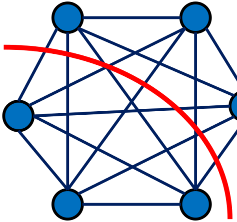
How many edges?

$|E_G| = \mathcal{O}(n^2)$

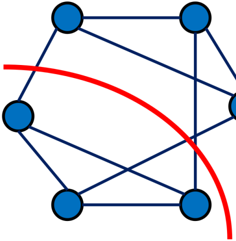
$|E_H| = \mathcal{O}(dn)$

Graph Sparsification: What is **good** sparse?

$G = K_n$



$H = d$-**regular** (random)



What are the cut weights for any S?

$w_G(\delta S) = |S| \cdot |\overline{S}|$ $\qquad$ $w_H(\delta S) \approx \frac{d}{n} \cdot |S| \cdot |\overline{S}|$

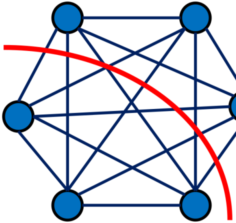
$$\forall S \subset V : \frac{w_G(\delta S)}{w_H(\delta S)} \approx \frac{n}{d}$$

Could be large :(

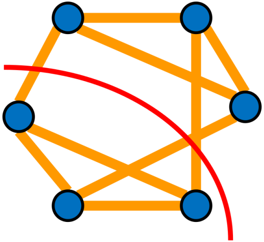
What to do?

Graph Sparsification: What is **good** sparse?

$G = K_n$



$H = d$-**regular** (random)



What are the cut weights for any S?

$$w_G(\delta S) = |S| \cdot |\overline{S}| \qquad w_H(\delta S) \approx \frac{d}{n} \cdot \frac{n}{d} \cdot |S| \cdot |\overline{S}|$$

$$\forall S \subset V : \frac{w_G(\delta S)}{w_H(\delta S)} \approx 1$$

Benczúr & Karger: Can find such H quickly for any G!

Graph Sparsification: What is good sparse?

Recall if $\mathbf{f} \in \{0,1\}^n$ represents S then $\mathbf{f}^\mathsf{T}\mathbf{L}_G\mathbf{f} = \mathrm{cut}_G(S)$

$$(1-\varepsilon)\mathrm{cut}_H(S) \le \mathrm{cut}_G(S) \le (1+\varepsilon)\mathrm{cut}_H(S)$$

becomes

$$(1-\varepsilon)\mathbf{f}^\mathsf{T}\mathbf{L}_H\mathbf{f} \le \mathbf{f}^\mathsf{T}\mathbf{L}_G\mathbf{f} \le (1+\varepsilon)\mathbf{f}^\mathsf{T}\mathbf{L}_H\mathbf{f}$$

If we ask this only for $\mathbf{f} \in \{0,1\}^n \to (1+\varepsilon)$-cut similar combinatorial

Benczúr & Karger (1996)

If we ask this for all $\mathbf{f} \in \mathbb{R}^n \to (1+\varepsilon)$-**spectrally similar**

Spielman & Teng (2004)

Spectral sparsifiers are stronger!

but checking for spectral similarity is easier

Spectral Graph Sparsification

Reason 1: Spectral sparsification helps when solving $\mathbf{L}_G\mathbf{x} = \mathbf{y}$

When a sparse H is spectrally similar to G then $\mathbf{x}^\mathsf{T}\mathbf{L}_G\mathbf{x} \approx \mathbf{x}^\mathsf{T}\mathbf{L}_H\mathbf{x}$

Gaussian Elimination	$\mathcal{O}(n^3)$
Fast Matrix Multiplication	$\mathcal{O}(n^{2.37})$
Spielman & Teng (2004)	$\mathcal{O}(m \log^{30} n)$
Koutis, Miller, and Peng (2010)	$\mathcal{O}(m \log n)$

Spectral Graph Sparsification

Reason 2: Spectral sparsification preserves eigenvalues!

Rayleigh-Ritz gives:

$$\lambda_{\min} = \min \frac{\mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x}}{\mathbf{x}^\mathsf{T}\mathbf{x}} \quad \text{and} \quad \lambda_{\max} = \max \frac{\mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x}}{\mathbf{x}^\mathsf{T}\mathbf{x}}$$

What can we say about $\lambda_i(G)$ and $\lambda_i(H)$?

$$(1-\varepsilon)\mathbf{f}^\mathsf{T}\mathbf{L}_G\mathbf{f} \leq \mathbf{f}^\mathsf{T}\mathbf{L}_H\mathbf{f} \leq (1+\varepsilon)\mathbf{f}^\mathsf{T}\mathbf{L}_G\mathbf{f}$$

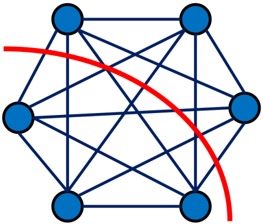
Eigenvalues are approximated well!

$$(1-\varepsilon)\lambda_i(G) \leq \lambda_i(H) \leq (1+\varepsilon)\lambda_i(G)$$

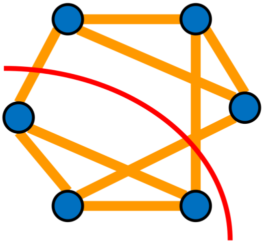
Other properties too: random walks, colorings, spanning trees, . . .

Spectral Graph Sparsification: Example

$G = K_n$



$H =$ **fat d-regular** (random)



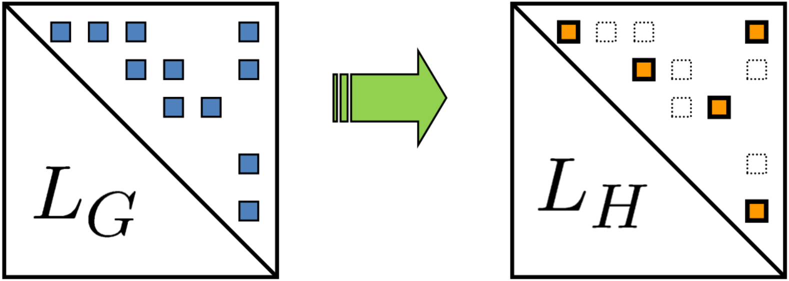
We wanted: $\forall S \subset V : \dfrac{w_G(\delta S)}{w_H(\delta S)} = \dfrac{\mathbf{x}_S^\mathsf{T}\mathbf{L}_G\mathbf{x}_S}{\mathbf{x}_S^\mathsf{T}\mathbf{L}_H\mathbf{x}_S} \approx 1 \pm \varepsilon$

Now we need: $\forall \mathbf{x} : \dfrac{\mathbf{x}^\mathsf{T}\mathbf{L}_G\mathbf{x}}{\mathbf{x}^\mathsf{T}\mathbf{L}_H\mathbf{x}} \approx 1 \pm \varepsilon$

To satisfy the condition: $d = \frac{1}{\varepsilon^2}$

Spectral Graph Sparsification

How to sparsify electrically? Given $\mathbf{L}_G$ find $\mathbf{L}_H$ ...



...such that $\mathbf{x}^\mathsf{T}\mathbf{L}_G\mathbf{x} \leq \mathbf{x}^\mathsf{T}\mathbf{L}_H\mathbf{x} \leq \kappa \cdot \mathbf{x}^\mathsf{T}\mathbf{L}_G\mathbf{x}$

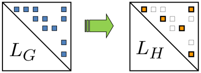
...we can also write $\mathbf{L}_G \preceq \mathbf{L}_H \preceq \kappa \cdot \mathbf{L}_G$

https://math.berkeley.edu/~nikhil/

Spectral Graph Sparsification

Let us consider unweighted graphs: $w_{ij} \in \{0, 1\}$

$$\mathbf{L}_G = \sum_{ij} w_{ij} \mathbf{L}_{ij} = \sum_{ij \in E} \mathbf{L}_{ij} = \sum_{ij \in E} (\boldsymbol{\delta}_i - \boldsymbol{\delta}_j)(\boldsymbol{\delta}_i - \boldsymbol{\delta}_j)^{\mathsf{T}} = \sum_{e \in E} \mathbf{b}_e \mathbf{b}_e^{\mathsf{T}}$$



We look for a **subgraph** H

$$\mathbf{L}_H = \sum_{e \in E} s_e \mathbf{b}_e \mathbf{b}_e^{\mathsf{T}} \quad \text{where } s_e \text{ is a new weight of edge e}$$

What **s** is good? | sparse! | Why would we want a subgraph?

Spectral Graph Sparsification

$$\text{We want} \quad \mathbf{L}_G \preceq \mathbf{L}_H \preceq \kappa \cdot \mathbf{L}_G$$

$$\text{That is, given } \mathbf{L}_G = \sum_{e \in E} \mathbf{b}_e \mathbf{b}_e^\mathsf{T} \text{ find } \mathbf{s}, \text{ s.t. } \mathbf{L}_G \preceq \sum_{e \in E} s_e \mathbf{b}_e \mathbf{b}_e^\mathsf{T} \preceq \kappa \cdot \mathbf{L}_G$$

$$\text{Forget } \mathbf{L}, \text{ given } \mathbf{V} = \sum_{e \in E} \mathbf{v}_e \mathbf{v}_e^\mathsf{T} \text{ find } \mathbf{s}, \text{ s.t. } \mathbf{V} \preceq \sum_{e \in E} s_e \mathbf{v}_e \mathbf{v}_e^\mathsf{T} \preceq \kappa \cdot \mathbf{V}$$

$$\text{Same as, given } \mathbf{I} = \sum_{e \in E} \mathbf{v}_e \mathbf{v}_e^\mathsf{T} \text{ find } \mathbf{s}, \text{ s.t. } \mathbf{I} \preceq \sum_{e \in E} s_e \mathbf{v}_e \mathbf{v}_e^\mathsf{T} \preceq \kappa \cdot \mathbf{I}$$

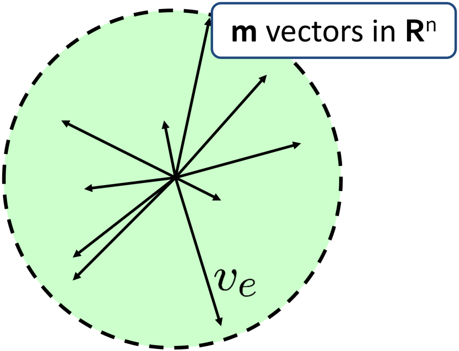
How to get it? $\mathbf{v}'_e \leftarrow \mathbf{V}^{-1/2} \mathbf{v}_e$

$$\text{Then } \textstyle\sum_{e \in E} s_e \mathbf{v}'_e (\mathbf{v}'_e)^\mathsf{T} \approx \mathbf{I} \iff \textstyle\sum_{e \in E} \mathbf{v}_e \mathbf{v}_e^\mathsf{T} \approx \mathbf{V}$$

multiplying by $\mathbf{V}^{1/2}$ on both sides

Spectral Graph Sparsification: Intuition

How does $\sum_{e \in E} \mathbf{v}_e \mathbf{v}_e^\mathsf{T} = \mathbf{I}$ look like geometrically?



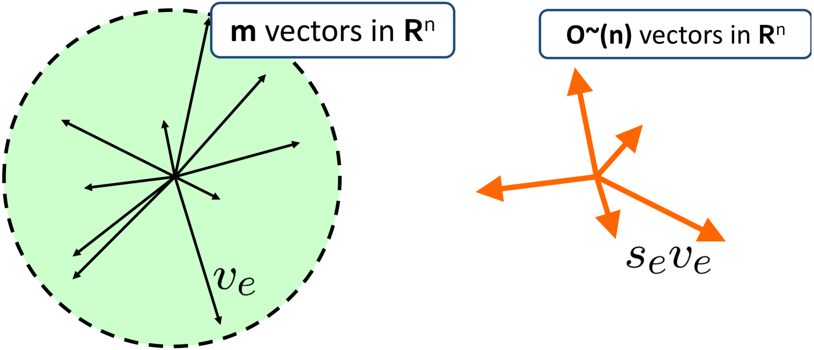
Decomposition of identity: $\forall \mathbf{u}$ (unit vector): $\sum_{e \in E} \mathbf{u}^\mathsf{T} \mathbf{v}_e = \mathbf{I}$

moment ellipse is a sphere

https://math.berkeley.edu/~nikhil/

Spectral Graph Sparsification: Intuition

What are we doing by choosing H?

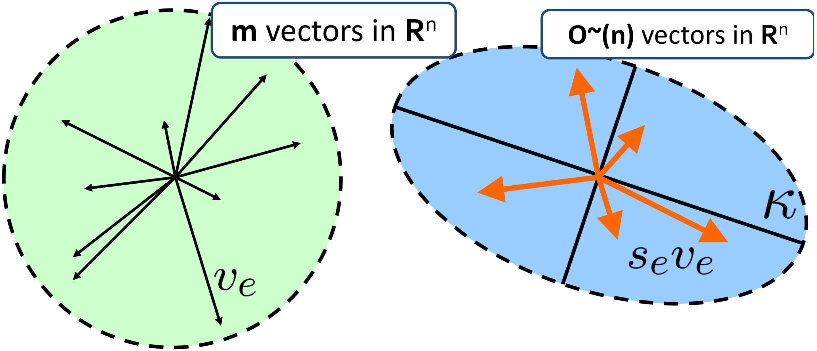


We take a subset of these $\mathbf{e}_e$s and scale them!

https://math.berkeley.edu/~nikhil/

Spectral Graph Sparsification: Intuition

What kind of scaling go we want?



Such that the blue ellipsoid looks like identity!

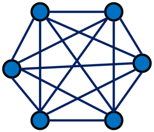
the blue eigenvalues are between 1 and κ

`https://math.berkeley.edu/~nikhil/`

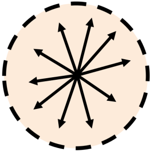
Spectral Graph Sparsification: Intuition

Example: What happens with K_n?

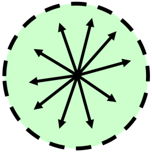
K_n graph



$\sum_{e \in E} \mathbf{b}_e \mathbf{b}_e^\mathsf{T} = \mathbf{L}_G$



$\sum_{e \in E} \mathbf{v}_e \mathbf{v}_e^\mathsf{T} = \mathbf{I}$



It is already isotropic! (looks like a sphere)

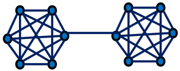
rescaling $\mathbf{v}_e = \mathbf{L}^{-1/2} \mathbf{b}_e$ does not change the shape

https://math.berkeley.edu/~nikhil/

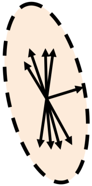
Spectral Graph Sparsification: Intuition

Example: What happens with a dumbbell?

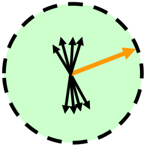
K_n graph



$\sum_{e \in E} \mathbf{b}_e \mathbf{b}_e^\mathsf{T} = \mathbf{L}_G$



$\sum_{e \in E} \mathbf{v}_e \mathbf{v}_e^\mathsf{T} = \mathbf{I}$



The vector corresponding to the link gets stretched!

because this transformation makes all the directions important

rescaling reveals the vectors that are critical

`https://math.berkeley.edu/~nikhil/`

Spectral Graph Sparsification: Intuition

What it this rescaling $\mathbf{v}_e = \mathbf{L}_G^{-1/2}\mathbf{b}_e$ doing to the norm?

$$\|\mathbf{v}_e\|^2 = \|\mathbf{L}_G^{-1/2}\mathbf{b}_e\|^2 = \mathbf{b}_e^{\mathsf{T}}\mathbf{L}_G^{-1}\mathbf{b}_e = R_{\text{eff}}(e)$$

reminder $R_{\text{eff}}(e)$ is the potential difference between the nodes when injecting a unit current

In other words: $R_{\text{eff}}(e)$ is related to the edge importance!

Electrical intuition: We want to find an electrically similar H and the importance of the edge is its effective resistance $R_{\text{eff}}(e)$.

Edges with higher R_{eff} are more **electrically significant**!

Spectral Graph Sparsification

Todo: Given $\mathbf{I} = \sum_e \mathbf{v}_e \mathbf{v}_e^\mathsf{T}$, find a sparse reweighting.

Randomized algorithm that finds $\mathbf{s}$:

- Sample $n \log n/\varepsilon^2$ with replacement $p_i \propto \|\mathbf{v}_e\|^2$ (resistances)
- Reweigh: $s_i = 1/p_i$ (to be unbiased)

Does this work?

Application of Matrix Chernoff Bound by Rudelson (1999)

$$1 - \varepsilon \prec \lambda \left(\sum_e s_e \mathbf{v}_e \mathbf{v}_e^\mathsf{T} \right) \prec 1 + \varepsilon$$

finer bounds now available

What is the the biggest problem here? Getting the p_is!

Spectral Graph Sparsification

We want to make this algorithm fast.

How can we compute the effective resistances?

$$\mathbf{L}_G = \sum_e \mathbf{b}_e \mathbf{b}_e^\mathsf{T} = \mathbf{B}^\mathsf{T}\mathbf{B} \qquad (\mathbf{B} \text{ has } \mathbf{b}_e^\mathsf{T}\text{s in rows -- } m \times n \text{ matrix})$$

$$\begin{aligned}\|\mathbf{v}_e\|^2 = p_i &= \mathbf{b}_e^\mathsf{T}\mathbf{L}_G^{-1}\mathbf{b}_e \\ &= \mathbf{b}_e^\mathsf{T}\mathbf{L}_G^{-1}\mathbf{B}^\mathsf{T}\mathbf{B}\mathbf{L}_G^{-1}\mathbf{b}_e \\ &= \|\mathbf{B}\mathbf{L}_G^{-1}(\delta_i - \delta_j)\|^2\end{aligned}$$

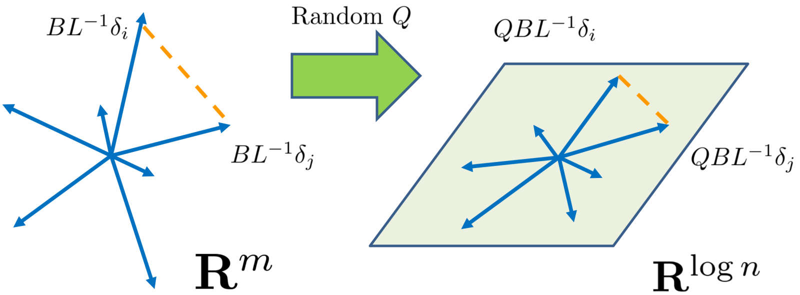
What does that mean?

It is a embedding of the distance (squared)!

Spectral Graph Sparsification

How to find a distance between the columns of a matrix $\mathbf{BL}_G^{-1}$?

$$R_{\text{eff}}(ij) = \|\mathbf{BL}_G^{-1}(\delta_i - \delta_j)\|^2$$



https://math.berkeley.edu/~nikhil/

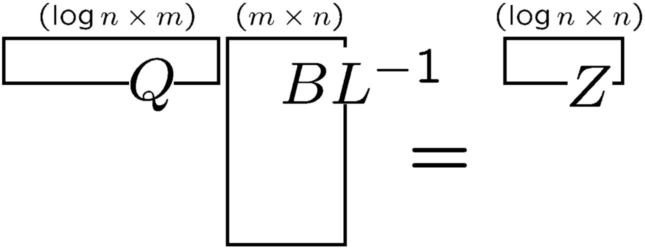
Spectral Graph Sparsification

How to find a distance between the columns of a matrix $\mathbf{BL}_G^{-1}$?

We never compute $\mathbf{BL}_G^{-1}$ we compute $\mathbf{QBL}_G^{-1}$!

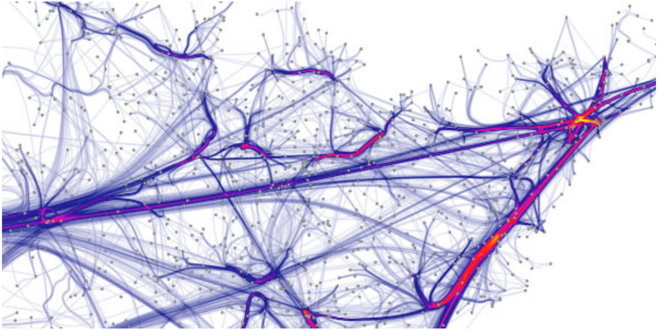
Johnson-Lindenstrauss: The distances are approximately preserved.

We take random $\mathbf{Q}_{\log n \times m}$ and set $\mathbf{Z} = \mathbf{QBL}_G^{-1}$



We solve $\mathcal{O}(\log n)$ (smaller) random linear systems!

Thank you!



Michal Valko

michal.valko@inria.fr

sequel.lille.inria.fr

SequeL – Inria Lille