Characterizing edges in signed and vector-valued graphs

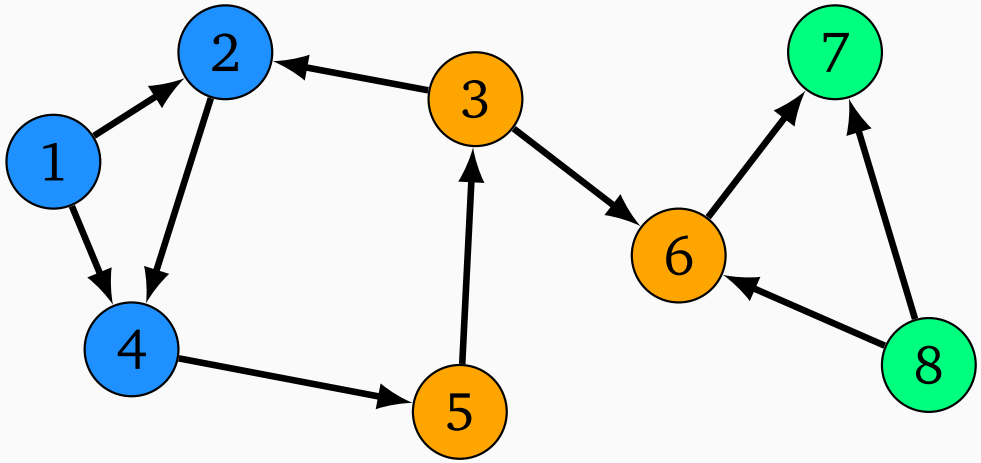
Géraud Le Falher

April 16, 2018

Introduction (8 minutes)

General context

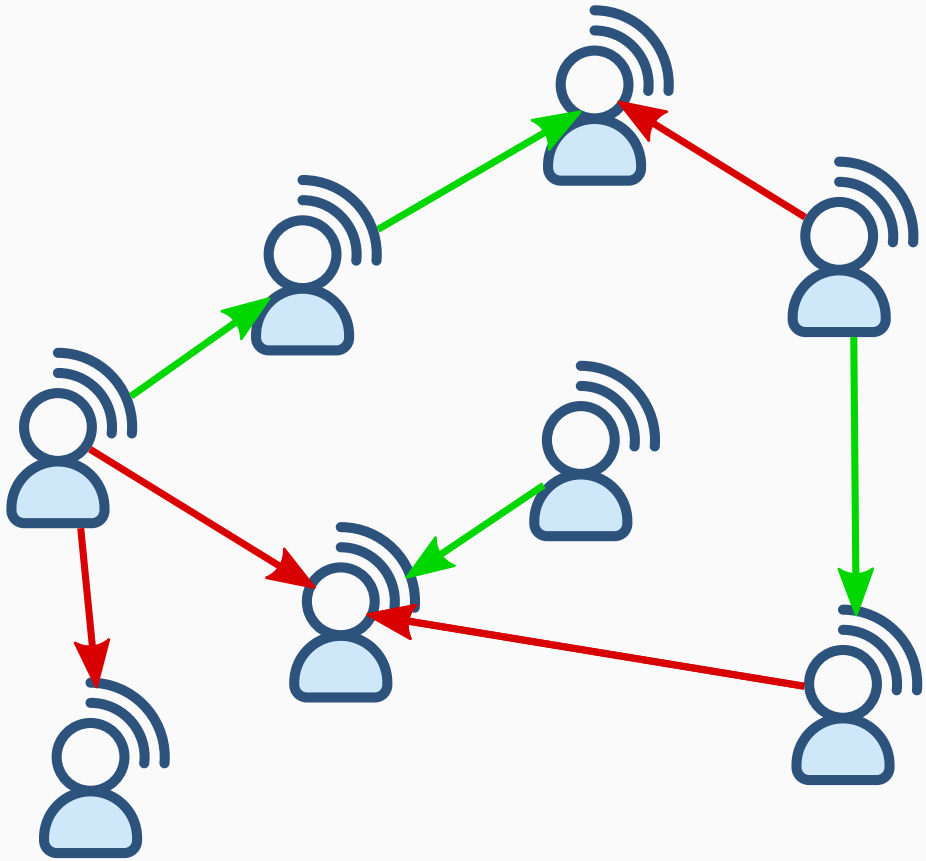
- In Machine Learning, we seek to automatically extract patterns from data and exploit them on future data.
- In this work, our data take the form of graphs, basically a set of entities (nodes) and their relationship (edges)



- Because of their simplicity and recent availability, graphs are ubiquitous, supporting tasks such as community detection, semi-supervised learning, link prediction and influence maximization.

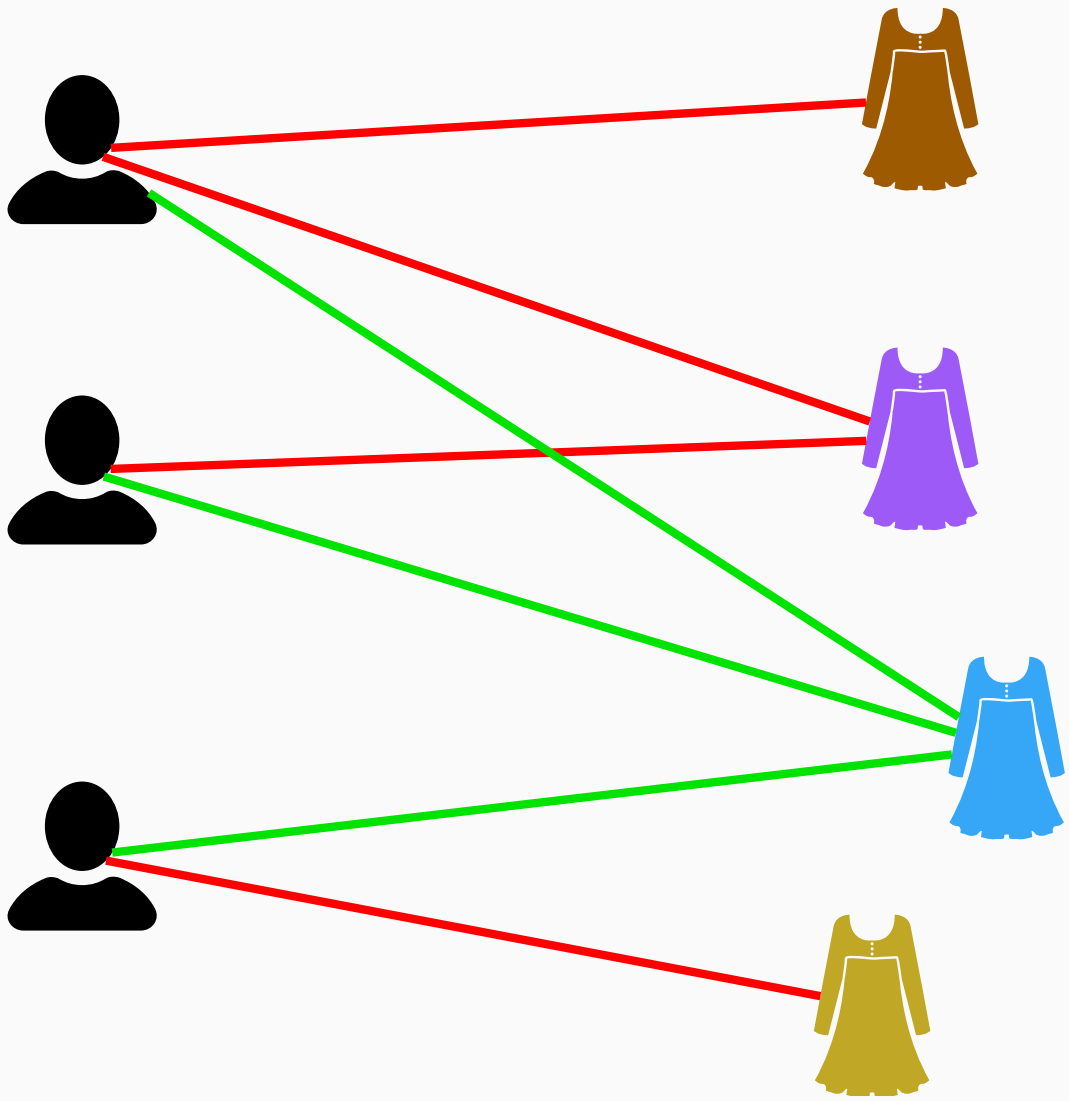
More complex graphs – Wikipedia

- Some graphs are more complex: multiple types of edge and nodes with attributes.
- Votes on Wikipedia: nodes are editors and edges represent for or against vote to get promoted to administrators



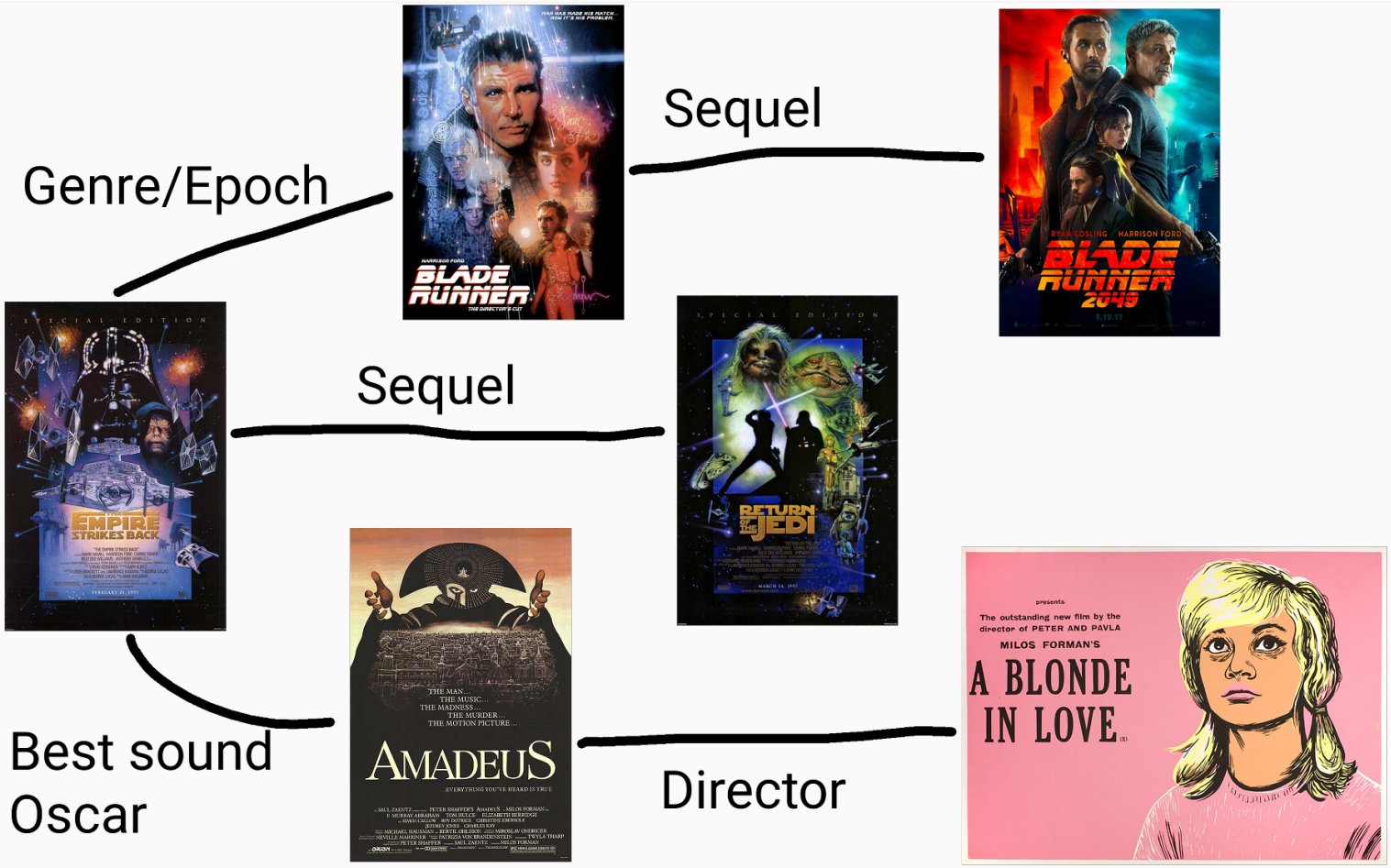
More complex graphs – Bipartite purchase

Bipartite purchase network: nodes are customers and products, edges are reviews



More complex graphs – Co-purchase

Attributed co-purchase network: nodes are products and their characteristic, edges denotes “frequently purchased together”



Thesis statement

- Such graphs are richer, but edge label might be costly to obtain, too numerous or missing.
- Fortunately, I will show that **there exist efficient and accurate methods to predict edge type in complex networks, relying only on the graph topology or also on node attributes.**

Outline

Introduction (8 minutes)

I - Directed signed graphs (8 mins)

II - Undirected signed graphs (8 minutes)

III - Node attributed graphs (15 minutes)

IV - Conclusion (6 minutes)

I - Directed signed graphs (8 mins)

Motivations

More Directed signed social networks (DSSN) besides Wikipedia editors:

Citations *i* cites the work of *j* to praise it or criticise it.

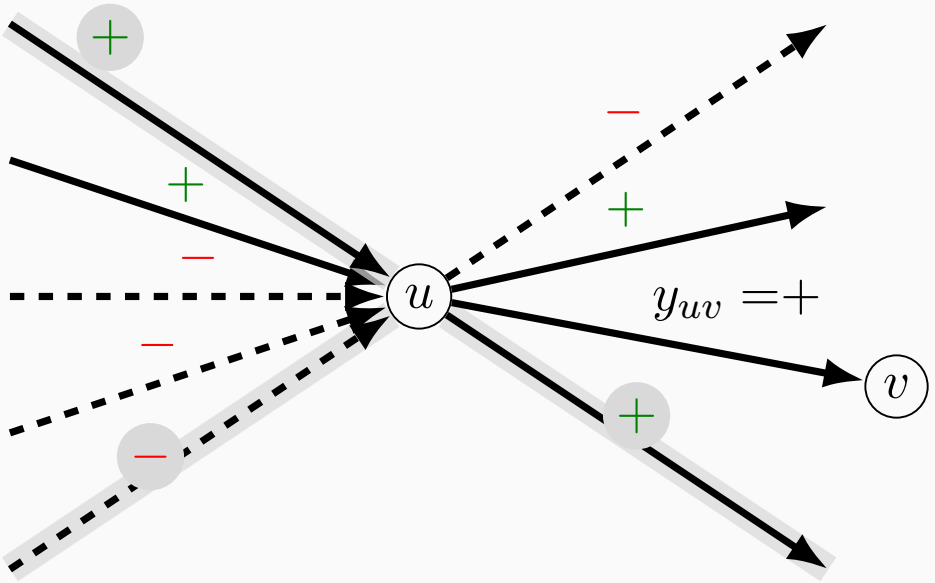
Slashdot *i* considers *j* as a friend or foe.

Epinion *i* trusts or not the reviews made by *j*.

Wik. Edits *i* reacted to a Wikipedia edit made by *j*, to enhance it or revert it.

Problem statement

Given the topology of a directed graph and the signs of some edges: predict the remaining signs $\rightarrow$ batch binary classification

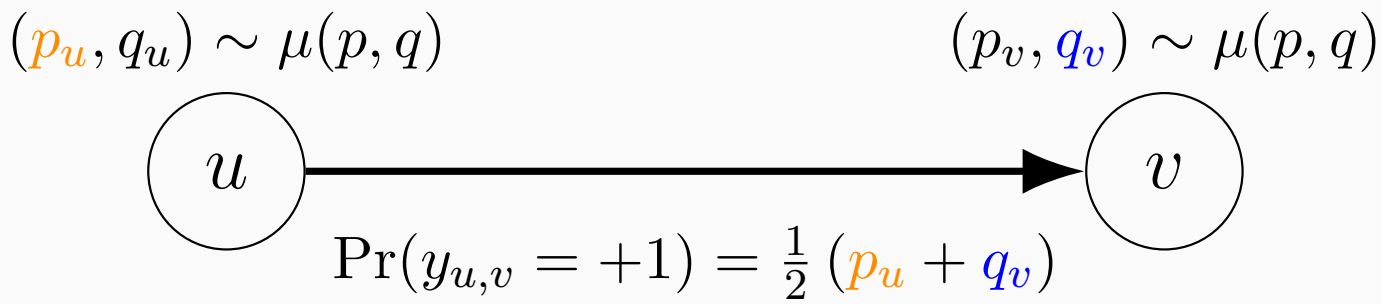


Being able to predict edge signs let us solve **practical, real world problems**:

- "Frenemy" detection
- Automatic moderation of large scale online interactions
- Cyber bullying prevention, at school or in online games

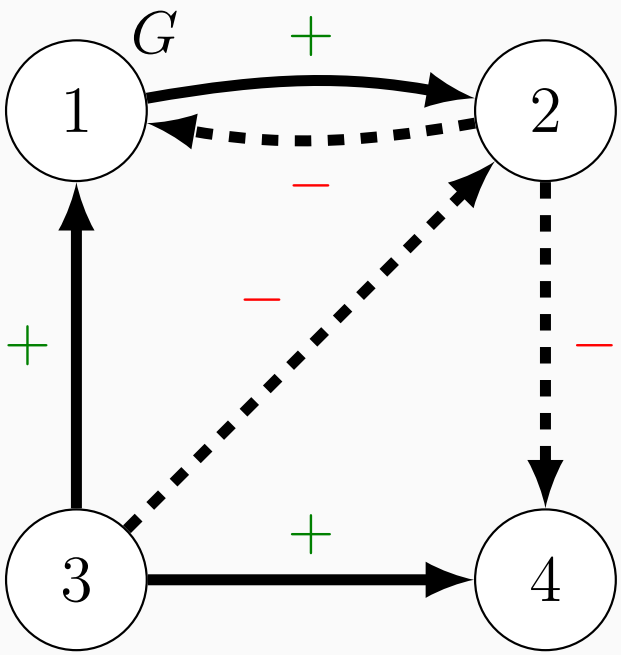
Solution Part 1: Generative model

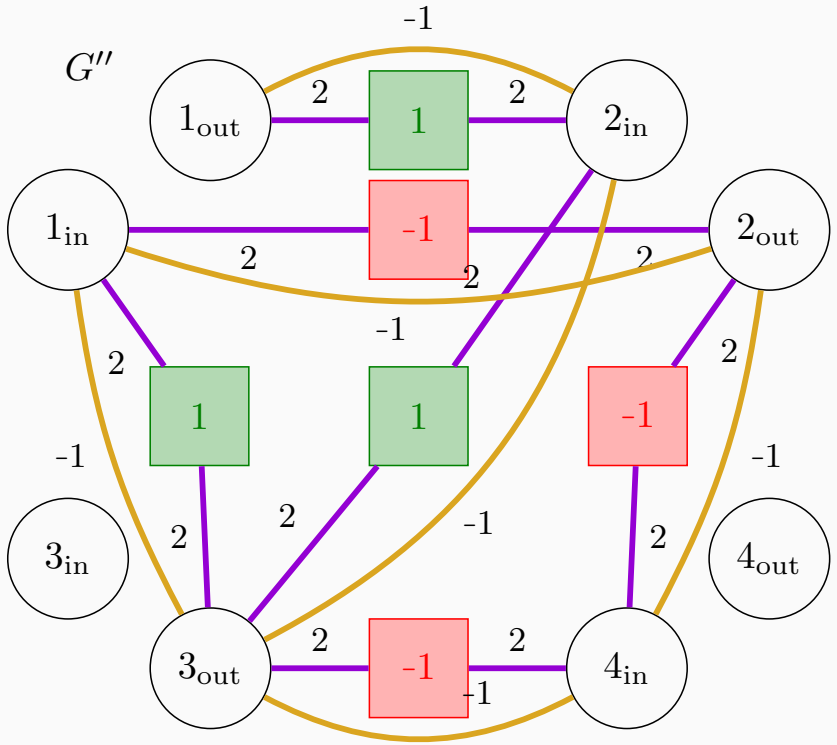
The first ingredient is a sign generative model, made of 2 parameters p and q governing emitting and receiving behavior and drawn from a prior distribution μ.



Solution Part 2: Graph transformation

The second one is a linear transformation of the graph, turning the problem into node classification. Doing label propagation on that new graph is an approximation of computing the maximum likelihood estimator of p and q.





Solution Part 3: Theoretical guarantees

Finally we show that when the graph is dense and the sampling uniform, directly estimating p and q on training data has good theoretical property, and performs well in practice anyway.

Namely, when we have $Q = \frac{1}{2\varepsilon^2} \ln \frac{4|V|}{\delta}$ samples of outgoing and incoming edges for every nodes, then, letting $\widehat{p}$, $\widehat{q}$ and $\widehat{\tau}$ being empirical estimates, we have that:

$$\left| \left[(1 - \widehat{p}_u) + (1 - \widehat{q}_v) - \widehat{\tau} \right] - \left[\frac{p_u + q_v}{2} \right] \right| \leq 8\varepsilon$$

holds with probability at least $1 - 10\delta$ simultaneously for all non-queried edges $(u, v) \in E$ such that $d_{\text{out}}(u), d_{\text{in}}(v) \geq Q$.

Experiments: Epinion dataset

Table 1: 100× MCC results on Epinion as $|E_0|$ grows

	Global	3%	9%	15%	20%	25%	time (ms)
LogReg		43.51	54.85	59.29	61.45	62.95	32
blc(tr, un)		41.39	53.23	57.76	60.06	61.93	**7**
L. Prop.	✓	**51.47**	**58.43**	**61.41**	**63.14**	**64.47**	1226
RankNodes	✓	**52.04**	**60.21**	**62.69**	**64.13**	**65.22**	2341
LowRank	✓	36.84	43.95	48.61	51.43	54.51	121530
Bayesian		31.00	48.24	56.88	61.49	64.45	116838
16 Triads		34.42	49.94	54.56	56.96	58.73	129

Experiments: Citations dataset

Table 2: 100× MCC results on Citations as $|E_0|$ grows

	Global	3%	9%	15%	20%	25%	time (ms)
LogReg		15.19	26.46	32.98	36.57	39.90	2
blc(tr, un)		15.09	26.40	32.98	36.72	40.16	**<1**
L. Prop.	✓	19.00	30.25	35.73	38.53	41.32	**16**
RankNodes	✓	12.28	24.44	31.03	34.57	38.26	**128**
LowRank	✓	8.85	17.08	22.57	25.57	29.24	1894
Bayesian		10.91	23.75	32.25	36.52	40.32	5398
16 Triads		8.62	16.42	22.01	24.77	27.13	5

Conclusion

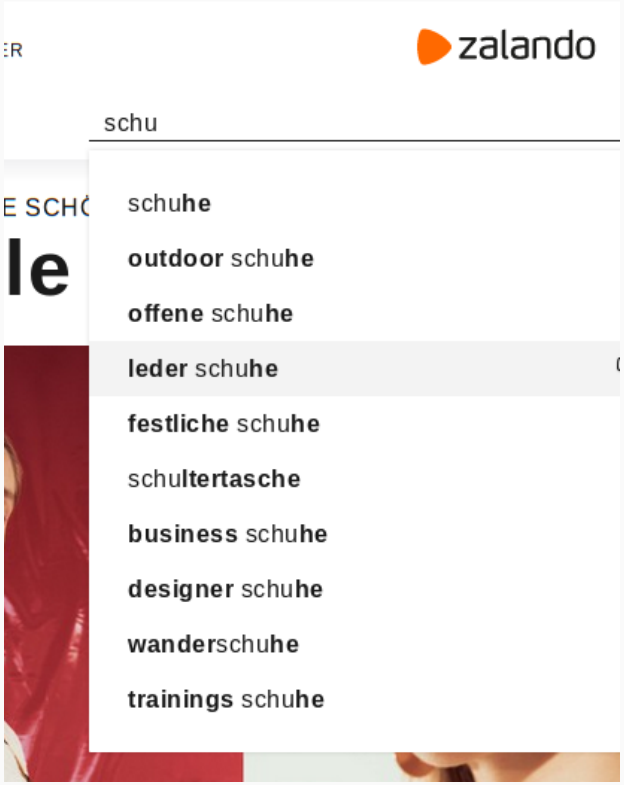
Because we only use the topology, our solution is both:

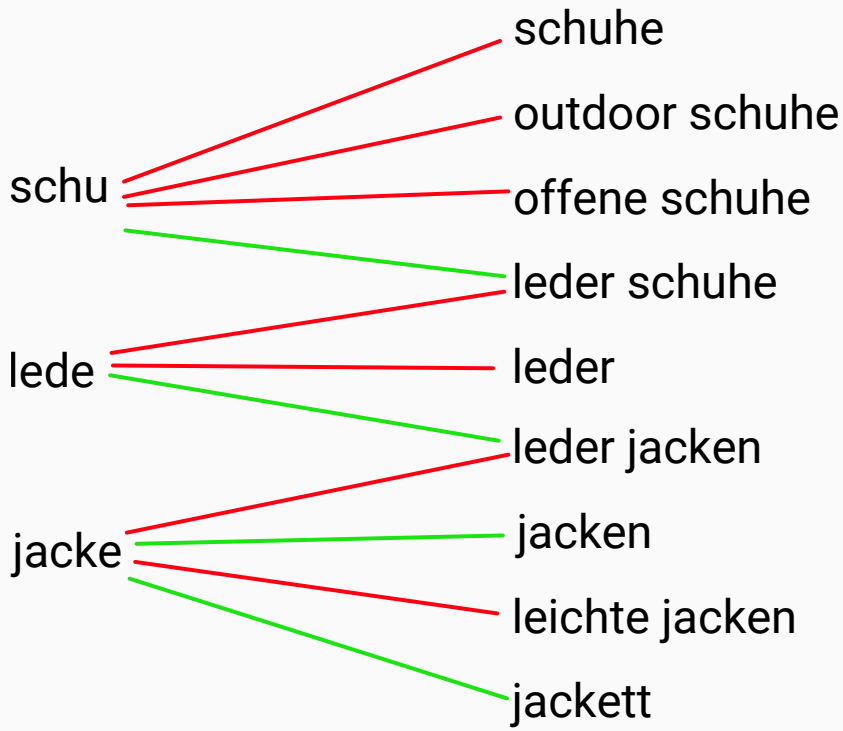
- domain agnostic
- fast

II - Undirected signed graphs (8 minutes)

Problem

- Our sign generative model is not suitable for undirected graphs. Yet such graphs are important, for instance for recommendation in bipartite graphs.

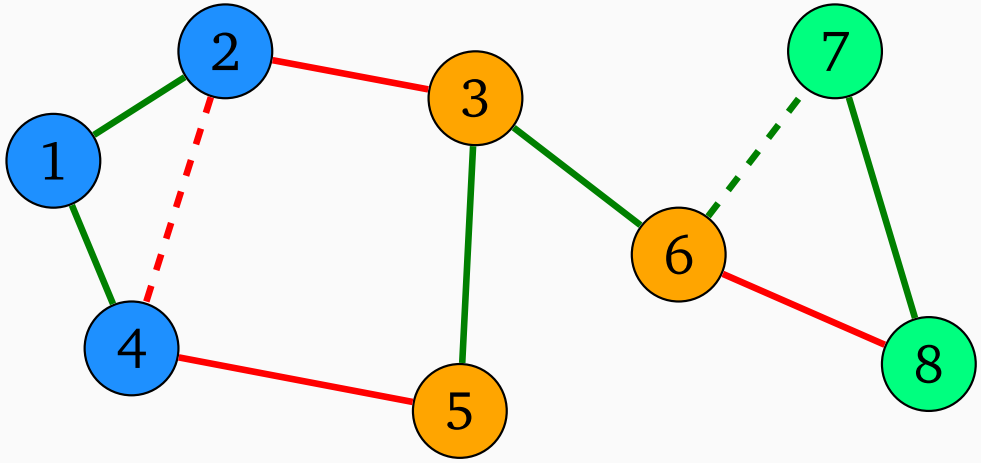




- We consider the active setting, where we first select a training set, query its signs and predict the remaining edges.

New bias

- Assume that nodes belongs to K latent groups, and that signs are governed by those groups, modulo some irregularities.



- This is motivated in social networks by the balance theory and in other graphs by assortative/dissortative patterns.
- Recovering those K groups in the presence of noise is the well studied Correlation Clustering problem (CC). The minimal objective value of CC when $K = 2$ is a bound on the number of mistakes for any active algorithms [1].

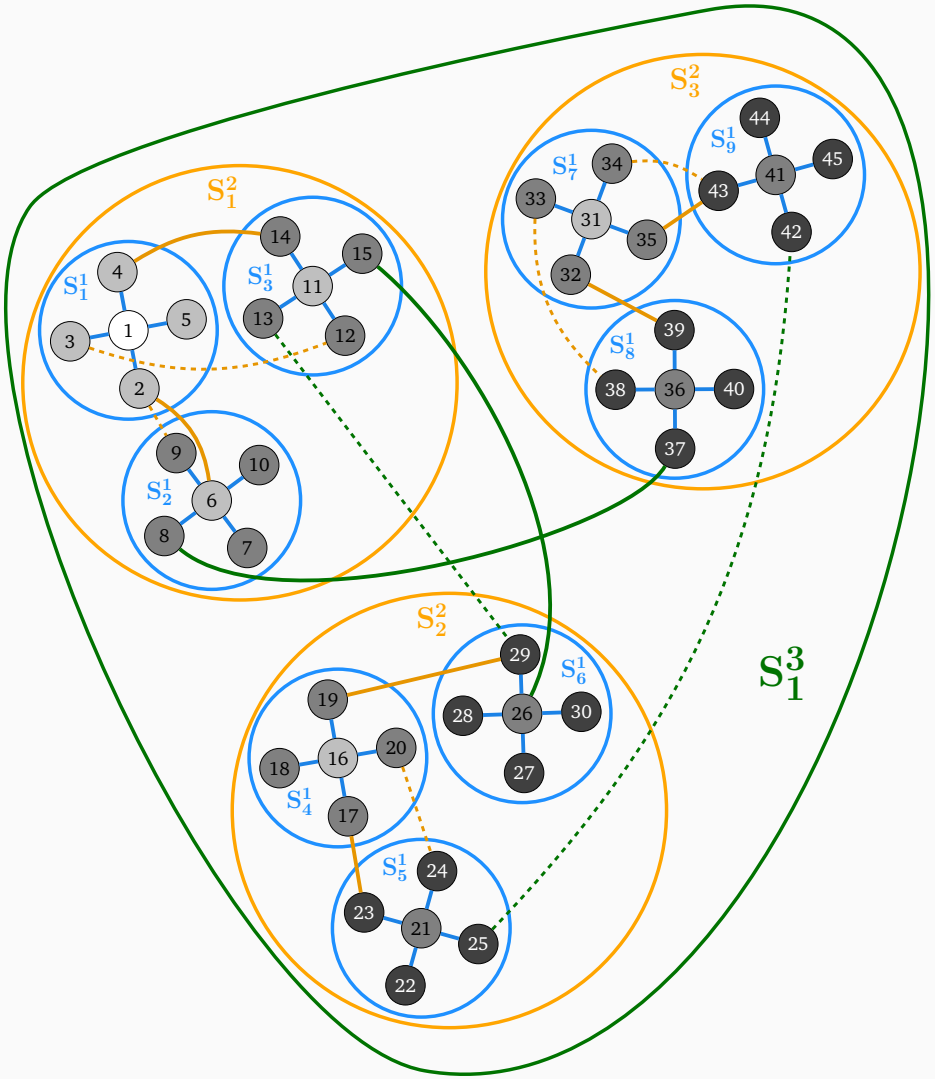
Solution: spanning structure

- An interesting case is $K = 2$ (strong balance), if there was no noise, the sign of any edge would the parity of any path between its endpoint.
- Since the noise is uniform (with probability q), the longer such path, the more likely we make an error by predicting using parity of observed edges. Overall, the expected number of errors is:

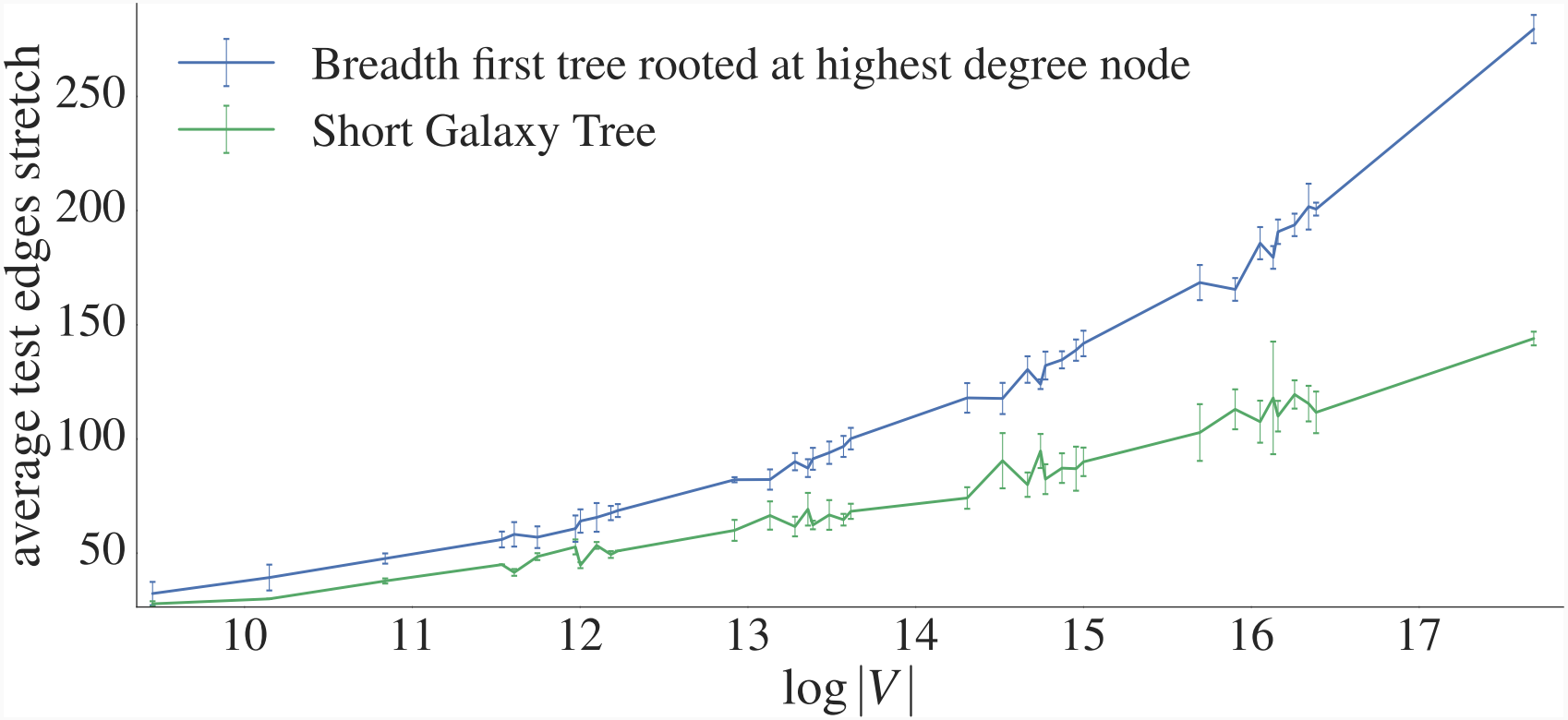
$$q\left(|E| + \sum_{(u,v)\in E_{\text{test}}} |\text{path}^T(u,v)|\right)$$

- Thus we look for a spanning structure with few edges that aims to minimize the stretch.

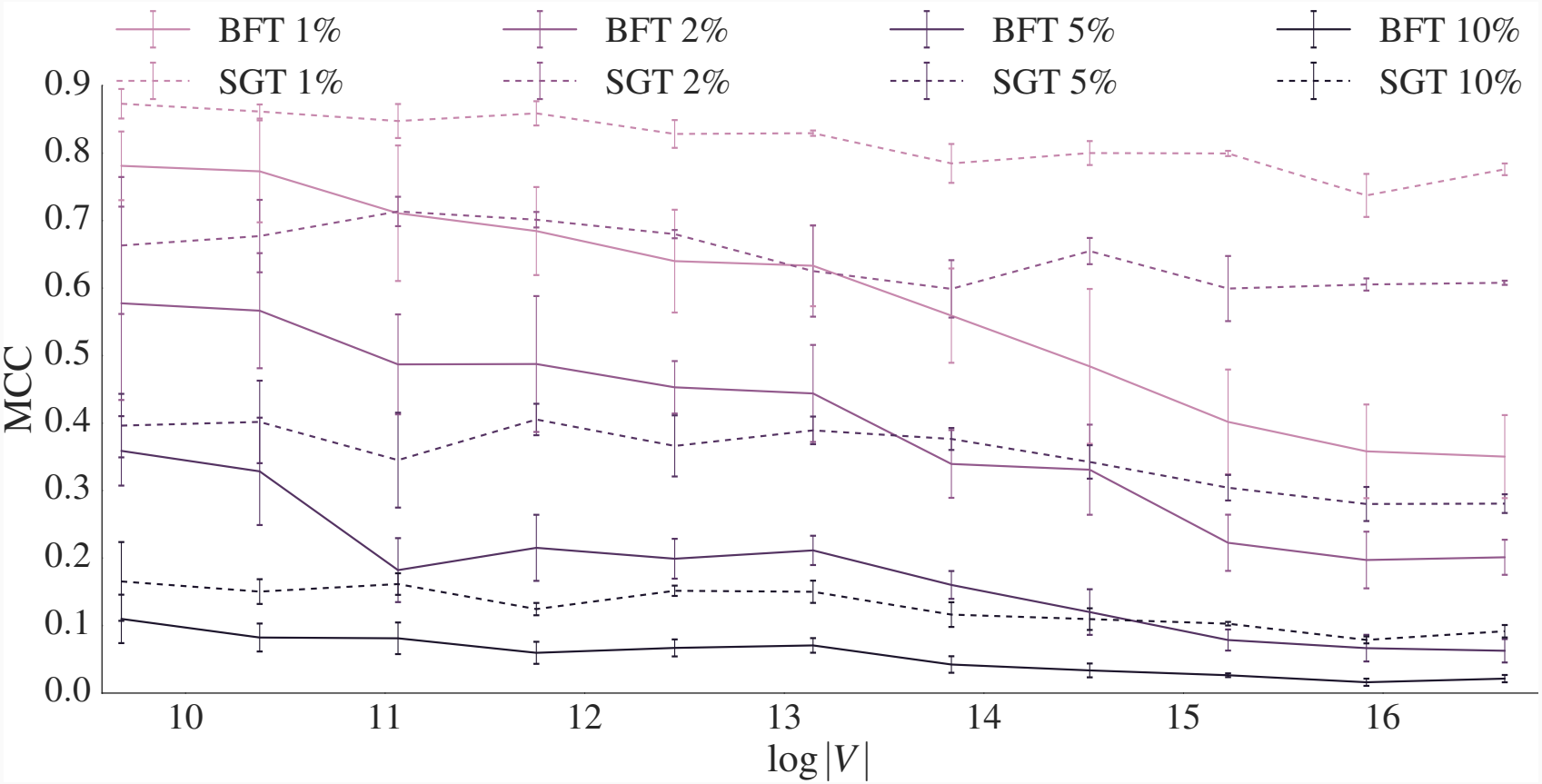
Construction



Experiments - Stretch



Experiments - Stretch



III - Node attributed graphs (15 minutes)

Problem

- Still characterizing edges, two big differences:
  - more than 2 types of edges $\rightarrow$ k-multilayer graphs
  - no label are provided at any point $\rightarrow$ unsupervised problem
- Additional information: node u has a profile $x_u \in [-1, 1]^d$
- Find k bounded "directions" and associate to every edge the "most explanatory" direction among those k

Motivations

- BlaBlaCar: nodes are users described by experience, location, preferences (music, smoke, talk, pet), reviews, while edges are shared rides. Those rides could be to go to work, cultural events, vacations and so on.
- Foursquare: nodes area venues described by location, reviews, category, time of visits, while edges are “frequently visited together” relation. Those venue groups can be part of party night, Sunday morning or sport routine.
- We model nuanced relations driven both by partial homophily and heterophily.

Formalization

- Combine the profiles of the edge u, v by entrywise product and score it with direction w_ℓ:

$$(x_u \circ x_v)^T w_\ell$$

- Associate the "most explanatory" direction to this edge:

$$\max_{\ell \in \{1,\dots,k\}} (x_u \circ x_v)^T w_\ell \quad \text{Let } w_\ell = w(u, v)$$

- Find the overall best k directions:

$$\arg\max_{\{w_1,\dots,w_k\} \subset \mathbb{B}^d} \sum_{u,v \in E} \max_{\ell \in \{1,\dots,k\}} (x_u \circ x_v)^T w_\ell$$

x_u	x_v	$x_u \circ x_v$	w_1	w_2
0.9	0.8	0.72	0.6	0.1
−0.8	−0.9	0.72	0.1	0.4
0.9	−0.8	−0.72	−0.7	0.6
0.0	0.5	0.0	0.1	0.4

$$(x_u \circ x_v)^T w_1 = 1.008$$
$$(x_u \circ x_v)^T w_2 = -0.072$$

Two Topological Constraints

1. The profile of each node is (close to) a linear combination of its incident directions: this introduces additional parameters ($\{a_{uv}\}_{u,v\in E}$ and $\{b_u\}_{u\in V}$) but provides more guidance.

$$\mathcal{L}_{\text{node}} = \sum_{u\in V} \left\| x_u - b_u - \sum_{v\in\mathcal{N}(u)} a_{uv} w(u,v) \right\|_2^2$$

2. Each node can only be involved in $k_{\text{local}} < k$ directions.

$$\mathcal{L}_{\text{local}} = \sum_{u\in V} \left\| \sum_{v\in\mathcal{N}(u)} y_{uv} \right\|_1$$

Think of y_{uv} as an indicator vector with k components. Its i^{th} component can be relaxed to

$$y_{uv;i} = \frac{\exp(\beta \left(x_u \circ x_v\right)^T w_i)}{\sum_{j=1}^{k} \exp(\beta \left(x_u \circ x_v\right)^T w_j)}$$

First Solutions

- A baseline is simply to cluster the $|E|$ $x_u \circ x_v$ in k clusters.
- It can be improved by plugging our scoring function in the Lloyd algorithm.
- Another approach is to directly optimize our objective function. We first make it convex by relaxing max using softmax.

$$\underset{\{w_1,\dots,w_k\}\subset\mathbb{B}^d}{\arg\max} \sum_{u,v\in E} \frac{1}{\beta} \sum_{\ell=1}^{k} \log\left(\exp\left(\beta\left(x_u \circ x_v\right)^T w_\ell\right)\right)$$

Then we take topology into account, by relaxing as well the two previous regularization terms, and optimize the difference of convex terms by gradient descent[1]

[1] Disciplined Convex-Concave Programming

Matrix Solution – Convex formulation

- Find a direction for each edge, but make sure those directions are linear combination of k "base" directions $\rightarrow$ low rank matrix.

$$\sum_{u,v \in E} \left(x_u \circ x_v\right)^T w_{uv} = \left\langle S^T, W \right\rangle_{\mathrm{F}}$$

$$\min_{W \in \mathbb{M}^{d \times |E|}} -\left\langle S^T, W \right\rangle_{\mathrm{F}} + \mathrm{rank}(W)$$

- Relax low rank by nuclear norm and ensure the norm of W columns are not too large. Optimized by the Frank Wolfe method, only requiring top singular vector at each iteration.

$$\min_{\substack{W \in \mathbb{R}^{d \times |E|} \\ \|W\|_* \leq \delta}} \left\langle \mu W - S^T, W \right\rangle_{\mathrm{F}}$$

Matrix Solution – Alternating formulation

- Write W explicitly as PQ^T and use alternating optimization.
- Those formulations have more parameters, but allow mixed membership.

Synthetic experiments

we generate data in order to have ground truth: (500 nodes, 1300 edges, 200 times k directions and corresponding profiles)

$\mathcal{D}_k$	k-means	Lloyd	Combined	Frank Wolfe	Explicit
default	.818	*.873**	.893*	.381	.893
$k = 5$	.836	*.838*	.875*	.213	.875
$k = 9$	.803	*.881**	.894*	.421	.894
$n_o = 6$	.813	*.824**	.856*	.378	.855
$n_o = 12$	*.827*	.823	.852*	.370	.851
$k_{\text{local}} = 4$	.772	*.814**	.853*	.320	.853
$d = 77$	.905	*.933**	.941*	.222	.931

Conclusions

- Given a graph with node profiles, find k directions explaining the existing connections
- linear model for performance and interpretability, two formulations:
  - difference of convex scalar functions
  - Matrix convex expression suitable for Frank–Wolfe algorithm
- Future work:
  - take time into account (homophily vs contagion)
  - multigraph, more than one type of edge between two nodes

IV - Conclusion (6 minutes)

Contribution

efficient and accurate methods to predict edge type in complex networks, relying only on the graph topology or also on node attributes.

	I	II	III
graph	directed, 2 types	undirected, 2 types	attributed, k types
learning setting	batch label	active label	no label
approach	estimate parameters	find spanning structure	matrix optimization
efficient	$O(\|E\|)$, parallel	$O(\|E\| \log \|E\|)$, conjectured	$O(dT\|E\|)$
accurate	close to Bayes predictor, efficient in practice	Close to Correlation Clustering bound	convex problem: global max (but no ground truth)

Future work

- Two ways relation with representation learning in graphs:
  - can give more accurate input to RL methods
  - can exploit RL features as node attributes
- *link prediction as Liva suggested?*

References

N. Cesa-Bianchi *et al.*, "A correlation clustering approach to link classification in signed networks," in *Proceedings of the 25th Annual Conference on Learning Theory*, vol. 23, 2012, pp. 1–20. [Online]. Available: http://jmlr.org/proceedings/papers/v23/cesa-bianchi12/cesa-bianchi12.pdf.

P. Auer *et al.*, "Adaptive and self-confident on-line learning algorithms," *Journal of Computer and System Sciences*, vol. 64, no. 1, pp. 48–75, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002200001917957.

Thank you!
Questions?

Setting

The signs are **adversarial** rather than generated by our model.

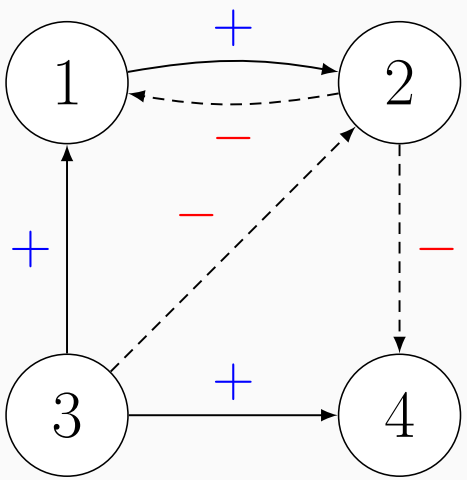
At each round, the learner is asked to predict one label, which is then revealed to him and the procedure repeats.

Labeling regularity

Letting Y be the vector of all labels, $\Psi_{\text{out}}(i, Y)$ is the number of least used label outgoing from i, and $\Psi_{\text{out}}(Y) = \sum_{i \in V} \Psi_{\text{out}}(i, Y)$.

Likewise for incoming edges, $\Psi_{\text{in}}(Y) = \sum_{j \in V} \Psi_{\text{in}}(j, Y)$ and finally $\Psi_G(Y) = \min\{\Psi_{\text{in}}(Y), \Psi_{\text{out}}(Y)\}$.

node i	1	2	3	4	
$\Psi_{\text{out}}(i, Y)$	0	0	1	0	$\Psi_{\text{out}}(Y) = 1$
$\Psi_{\text{in}}(i, Y)$	1	0	0	1	$\Psi_{\text{in}}(Y) = 2$



Online algorithm and bounds

- Our algorithm is a combination of Randomized Weighted Majority instances built on top of each other.

Online algorithm and bounds

- Our algorithm is a combination of Randomized Weighted Majority instances built on top of each other.
- On average, it makes $\Psi_G(Y) + O\left(\sqrt{|V|\Psi_G(Y)} + |V|\right)$ mistakes.

Online algorithm and bounds

- Our algorithm is a combination of Randomized Weighted Majority instances built on top of each other.
- On average, it makes $\Psi_G(Y) + O\left(\sqrt{|V|\Psi_G(Y)} + |V|\right)$ mistakes.
- On the lower side, for any directed graph G and any integer K, there exists a labeling Y forcing at least $\frac{K}{2}$ mistakes to any online algorithms, while $\Psi_G(Y) \leq K$.

Online algorithm, 1. RWM node instances

For each node i, we predict the sign of edge outgoing from i by relying on two constant experts, always predicting -1 or always predicting $+1$. The best one will make $\Psi_{\mathrm{out}}(i, Y)$ mistakes. We combine them in a Randomized Weighted Majority algorithm (RWM) instance associated with i, call it $RWM_{out}(i)$. The instance expected number of mistakes is therefore [2], denoting by $M(i, j)$ the indicator function of a mistake on edge (i, j)

$$\sum_{j \in \mathcal{E}_{\mathrm{out}}(i)} \mathbb{E}\, M(i, j) = \Psi_{\mathrm{out}}(i, Y) + O\left(\sqrt{\Psi_{\mathrm{out}}(i, Y)} + 1\right)$$

We use the same technique to predict incoming edges of each node j, the instance $RWM_{in}(j)$ having the following average number of mistakes

$$\sum_{i \in \mathcal{E}_{\mathrm{in}}(j)} \mathbb{E}\, M(i, j) = \Psi_{\mathrm{in}}(j, Y) + O\left(\sqrt{\Psi_{\mathrm{in}}(j, Y)} + 1\right)$$

Online algorithm, 2. combining instances

We then define two meta experts: RWM_{out}, which predicts $y_{i,j}$ as $RWM_{out}(i)$, and RWM_{in}, which predicts $y_{i,j}$ as $RWM_{in}(j)$. Summing over all nodes, the number of mistakes of these two experts satisfy

$$\sum_{i \in V} \sum_{j \in \mathcal{E}_{\text{out}}(i)} \mathbb{E}\, M(i,j) = \Psi_{\text{out}}(Y) + O\left(\sqrt{|V|\Psi_{\text{out}}(Y)} + |V|\right)$$

$$\sum_{j \in V} \sum_{i \in \mathcal{E}_{\text{in}}(j)} \mathbb{E}\, M(i,j) = \Psi_{\text{in}}(Y) + O\left(\sqrt{|V|\Psi_{\text{in}}(Y)} + |V|\right)$$

Online algorithm, 3. final prediction

Our final predictor is a RWM combination of RWM_{out} and RWM_{out}, whose expected number of mistakes is

$$\sum_{(i,j)\in E} \mathbb{E}\, M(i,j) = \Psi_G(Y) + O\Bigg(\sqrt{|V|\Psi_G(Y)} + |V| + \sqrt{\left(\Psi_G(Y) + |V| + \sqrt{|V|\Psi_G(Y)}\right)} \Bigg)$$

$$= \Psi_G(Y) + O\left(\sqrt{|V|\Psi_G(Y)} + |V| \right)$$

Datasets properties

Table 3: Dataset properties.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$\frac{\lvert E\rvert}{\lvert V\rvert}$	$\frac{\lvert E^+\rvert}{\lvert E\rvert}$	$\frac{\Psi_{G''}(Y)}{\lvert E\rvert}$	$\frac{\Psi_G(Y)}{\lvert E\rvert}$
Citations	4 831	39 452	8.1	72.33%	.076	.191
Wikipedia	7 114	103 108	14.5	78.79%	.063	.142
Slashdot	82 140	549 202	6.7	77.40%	.059	.143
Epinion	131 580	840 799	6.4	85.29%	.031	.074
Wik. Edits	138 587	740 106	5.3	87.89%	.034	.086

$$\Psi_{G''}(Y) = \min_{\boldsymbol{p},\boldsymbol{q}\in[0,1]^{|V|}} \sum_{(i,j)\in E} \left(\frac{1+y_{i,j}}{2} - \frac{p_i+q_j}{2} \right)^2$$

Data generation

- create a random graph and pick for each a set of k_{local} directions.
- pick for each edge (u, v) a direction index $y_{uv} \in [\![k]\!]$ among the ones shared by its endpoint.
- draw the k directions at random
- optimize the profiles so as to maximize the edges goodness, minimize the term $\mathcal{L}_{\text{node}}$ and enforces as much as possible that for every edge $(u, v) \in E$, $\mathcal{E}(u, v) = y_{uv}$.

(take a look a Liva comment about Lagrangian relaxation of integer program...)