Adaptive Techniques for Learning over Graphs

PhD Final Oral Exam

Dimitris Berberidis

Dept. of ECE and Digital Tech. Center, University of Minnesota

Acknowledgements: Profs G. B. Giannakis, G. Karypis, Z. Zhang, and M. Hong

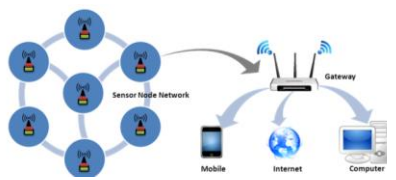
UNIVERSITY OF MINNESOTA
Driven to Discover℠

Minneapolis, Jan. 25, 2019

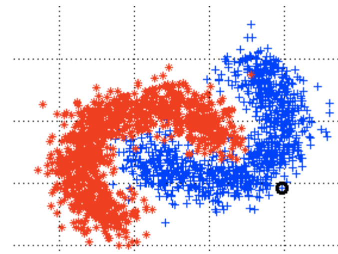
Motivation

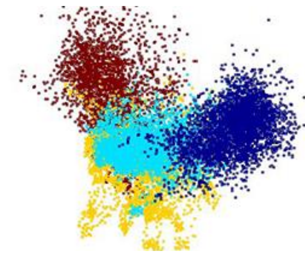
Graph representations

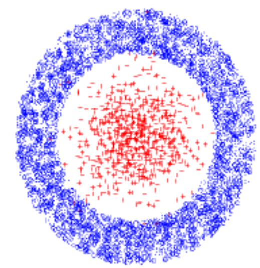
Real networks

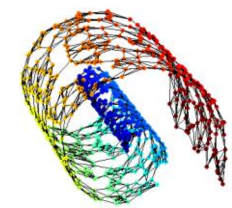


Data similarities



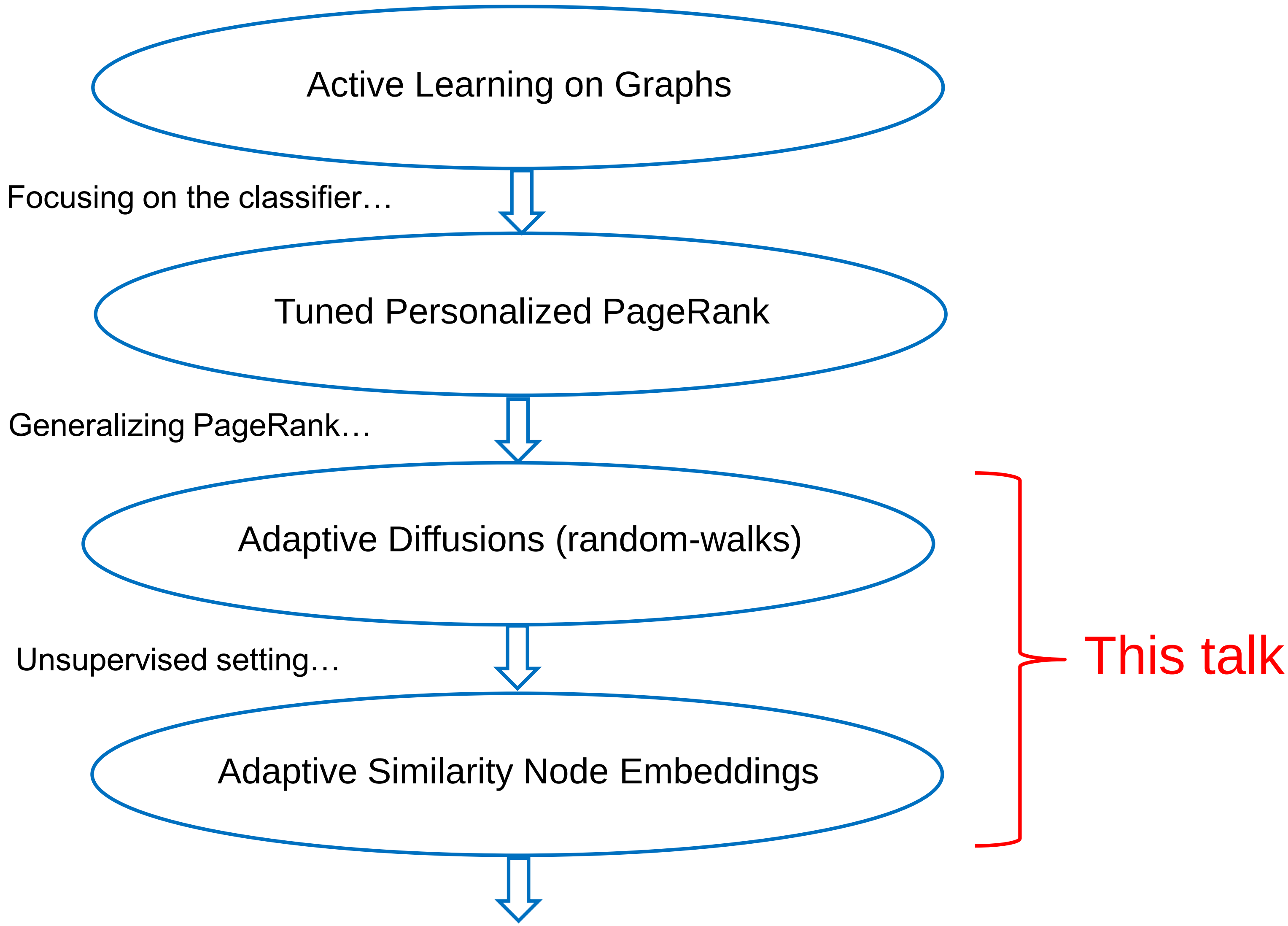






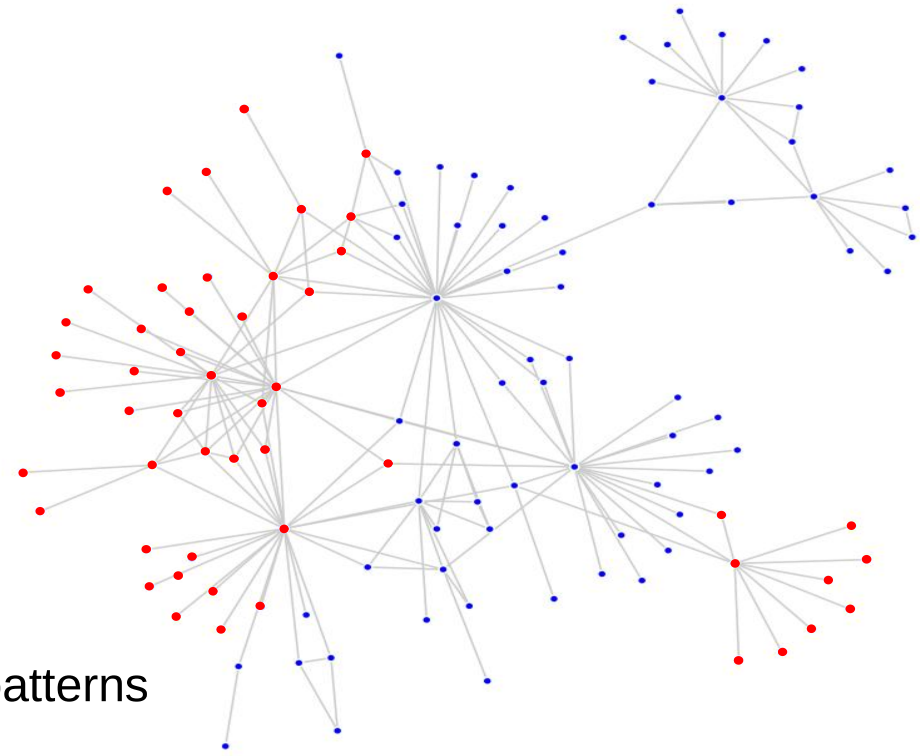
- **Objectives**: Learn-over/ mine/ manipulate real world graphs
- **Challenges**
  - Graphs can be **huge** with **few/none/unreliable labels** available
  - Graphs from different sources may have different properties

Roadmap-Timeline



Semi-supervised node classification

- Graph $\mathcal{G} := \{\mathcal{V}, \mathcal{E}\}$
  - Weighted adjacency matrix $\mathbf{A}$
  - Label $y_i \in \mathcal{Y}$ per node v_i
- Topology given or identifiable
- Main assumption
  - Graph topology relevant to label patterns

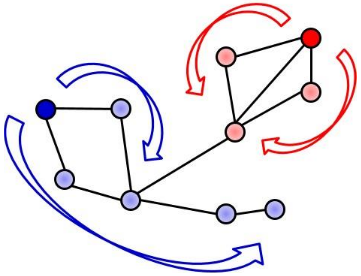


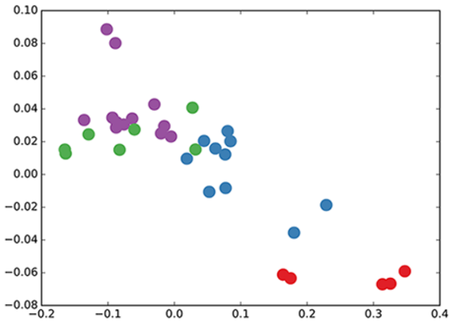
Goal: Given labels on $\mathcal{L} \subseteq \mathcal{V}$ learn unlabeled nodes $\mathcal{U} := \mathcal{V} \setminus \mathcal{L}$

Work in context

- Non-parametric semi-supervised learning (SSL) on graphs
  - Graph partitioning [Joachims et al '03]
  - Manifold regularization [Belkin et al '06]
  - Label propagation [Zhu et al'03, Bengio et al'06]
  - Bootstrapped label propagation [Cohen'17]
  - Competitive infection models [Rosenfeld'17]

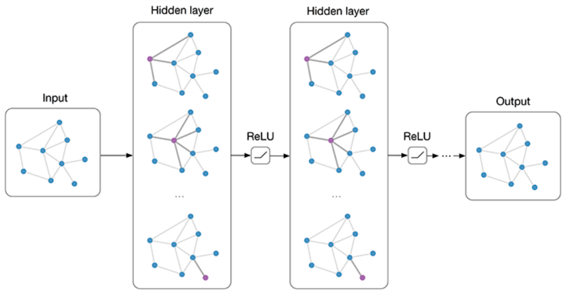
Label propagation





- Node embedding + classification of vectors
  - Node2vec [Grover et al '16]
  - Planetoid [Yang et al '16]
  - Deepwalk [Perozzi et al '14]

- Graph convolutional networks (GCNs)
  - [Atwood et al '16], [Kipf et al '16]



Random walks for SSL

- Consider a Random Walk on $\mathcal{G} := \{\mathcal{V}, \mathcal{E}\}$ with transition matrix $\mathbf{H} := \mathbf{A}\mathbf{D}^{-1}$.
- **K-step "landing" prob.** of a walk "rooted" on the labeled nodes of each class.

$$\mathbf{p}_c^{(k)} = \mathbf{H}^k \mathbf{v}_c \qquad [\mathbf{v}_c]_i = \begin{cases} 1/|\mathcal{L}_c|, & i \in \mathcal{L}_c \\ 0, & \text{else} \end{cases}$$

$$\mathcal{L}_c := \{i \in \mathcal{L} : y_i = c\}$$

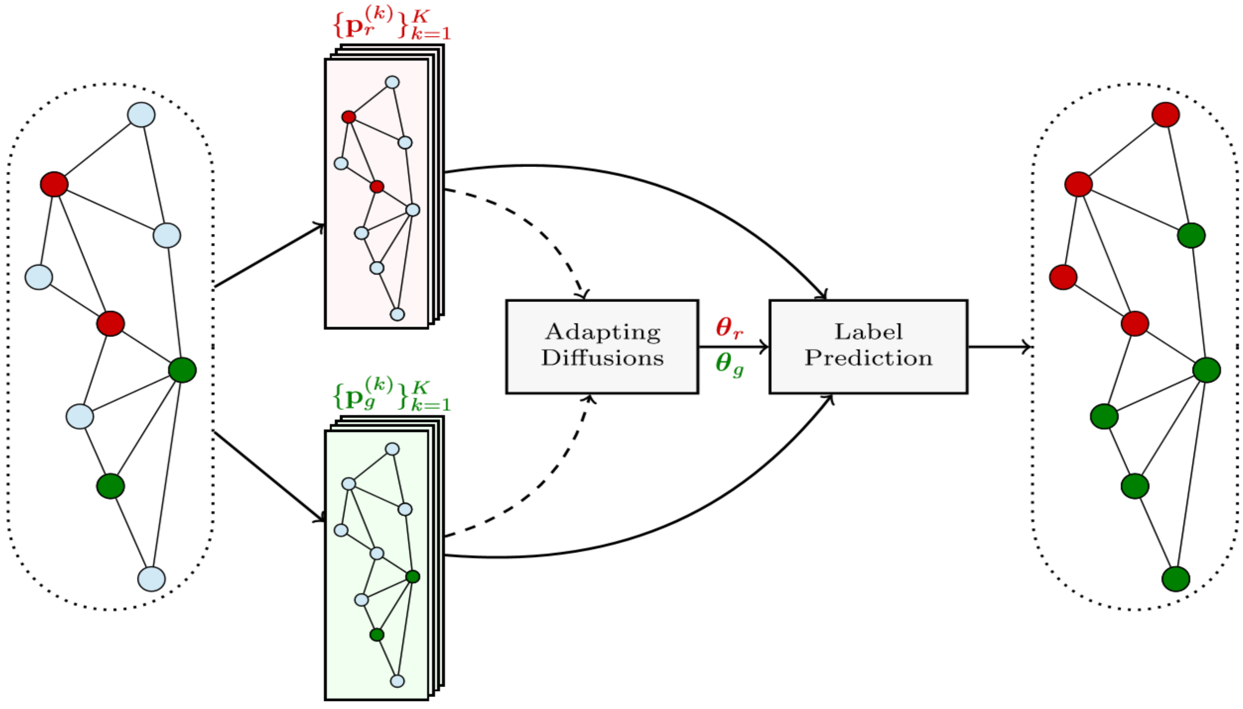
- Use the landing probabilities to create an **"influence" vector** for each class

$$\mathbf{f}_c(\boldsymbol{\theta}) := \sum_{k=1}^{K} \theta_k \mathbf{p}_c^{(k)} = \mathbf{P}_c^{(K)} \boldsymbol{\theta},$$

- Classify the unlabeled nodes as $\hat{y}_i(\boldsymbol{\theta}) := \arg\max_{c \in \mathcal{Y}} [\mathbf{f}_c(\boldsymbol{\theta})]_i$
- **Fixed θ:** Pers. PageRank (PPR) [Lin'10] , Heat kernel (HK) [Chung'07]

Our contribution: Graph- and label-adaptive selection of $\boldsymbol{\theta}_c$

AdaDIF



$$\hat{\boldsymbol{\theta}}_c = \arg\min_{\boldsymbol{\theta}\in\mathcal{S}^K} \ell(\mathbf{y}_{\mathcal{L}_c}, \mathbf{f}_c(\boldsymbol{\theta})) + \lambda R(\mathbf{f}_c(\boldsymbol{\theta}))$$

$$\mathcal{S}^K := \{\boldsymbol{\theta} \in \mathbb{R}^K : \ \boldsymbol{\theta} \geq \mathbf{0}, \ \mathbf{1}^\top \boldsymbol{\theta} = 1\}$$

Normalized label indicator vector

$$\ell(\mathbf{y}_{\mathcal{L}_c}, \mathbf{f}) = \sum_{i\in\mathcal{L}} \frac{1}{d_i}(y_i - f_i)^2 = (\bar{\mathbf{y}}_{\mathcal{L}_c} - \mathbf{f})^\top \mathbf{D}_{\mathcal{L}}^{-1}(\bar{\mathbf{y}}_{\mathcal{L}_c} - \mathbf{f})$$

$$R(\mathbf{f}) = \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left(\frac{f_i}{d_i} - \frac{f_j}{d_j}\right)^2 = \mathbf{f}^\top \mathbf{D}^{-1}\mathbf{L}\mathbf{D}^{-1}\mathbf{f}$$

AdaDIF complexity and the choice of K

- Complexity **linear** in nnz(**H**) and **quadratic** in K.

Theorem For any diffusion-based classifier with coefficients constrained to a probability simplex of appropriate dimensions, it holds that

$$K_\gamma \leq \frac{1}{\mu'} \log\left[\frac{2\sqrt{d_{\max}}}{\gamma}\left(\sqrt{\frac{1}{d_{\min_-}|\mathcal{L}_-|}} + \sqrt{\frac{1}{d_{\min_+}|\mathcal{L}_+|}}\right)\right]$$

where $d_{\min+} := \min_{i \in \mathcal{L}_+} d_i,\ d_{\min-} := \min_{j \in \mathcal{L}_-} d_j,\ d_{\max} := \max_{i \in \mathcal{V}} d_i \quad \mu' := \min\{\mu_2, 2-\mu_N\},$

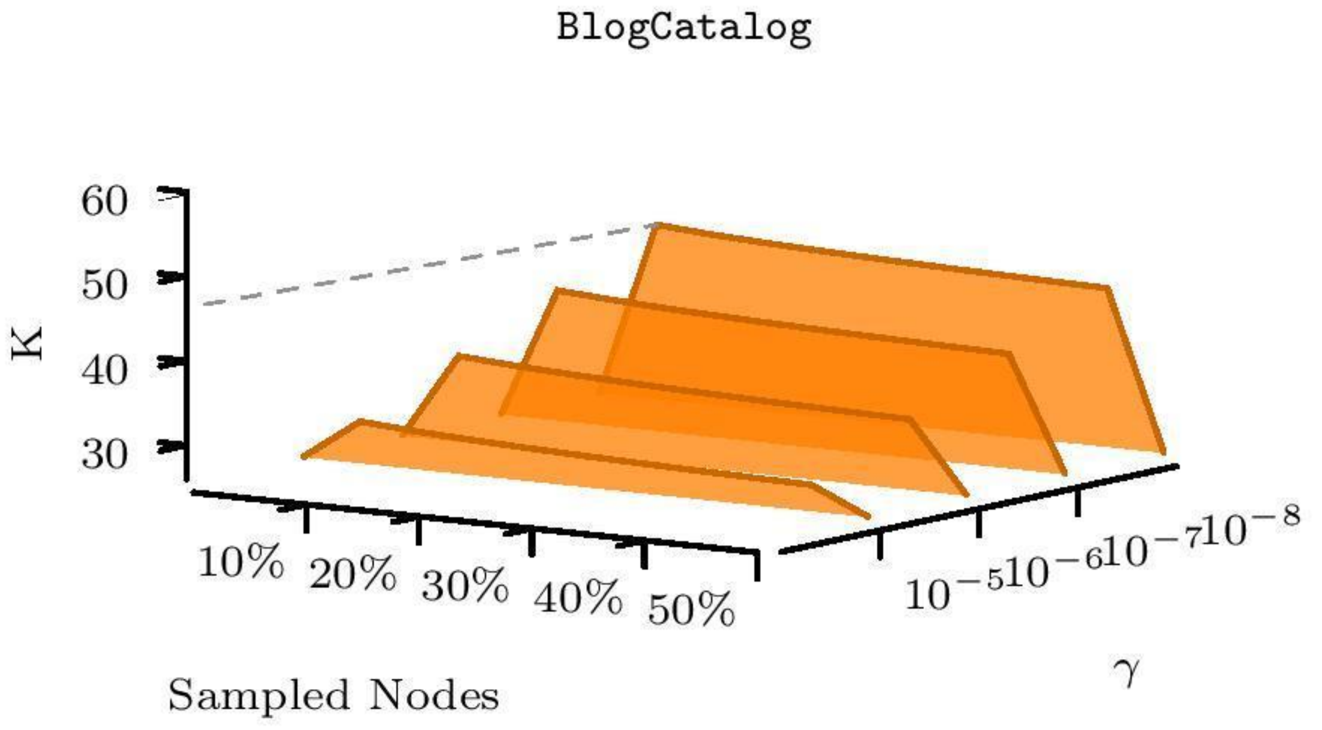
with $\{\mu_n\}_{n=1}^N$ the eigenvalues of the normalized graph Laplacian in ascending order.

- **Main message**:
  - Increasing K does not help distinguishing between classes
  - For most graphs a very small K suffices → AdaDIF will be very efficient!
  - If K needs to be large: **Dictionary of Diffusions** $\mathbf{C} := [\mathbf{c}_1 \cdots \mathbf{c}_D] \in \mathbb{R}^{K \times D}$

$$\mathbf{f}_c(\boldsymbol{\theta}) = \sum_{k=1}^K a_k(\theta)\mathbf{p}_c^{(k)} = \mathbf{P}_c^{(K)}\mathbf{a}(\boldsymbol{\theta}) = \mathbf{P}_c^{(K)}\,\mathbf{C}\boldsymbol{\theta}$$

  - Trading flexibility for complexity **linear** in both **nnz(H)** and K

Bound in practice



Real data tests

Competing baselines

- DeepWalk, Node2vec
- Planetoid, GCNN
- HK, PPR, Label Prop. (LP)

Graph	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{Y}\rvert$	Multi-label
Citeseer	3,233	9,464	6	No
Cora	2,708	10,858	7	No
PubMed	19,717	88,676	3	No
PPI (H. Sapiens)	3,890	76,584	50	Yes
Wikipedia	4,733	184,182	40	Yes
BlogCatalog	10,312	333,983	39	Yes

Evaluation metrics

$$\text{F1 score} = 2 \cdot \frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}} = \frac{2\cdot\text{true positive}}{2\cdot\text{true positive}+\text{false positive}+\text{false negative}}$$

- Micro-F1: node-centric accuracy measure
- Macro-F1: class-centric accuracy measure

- Cross-validation for PPR (α), HK (t), Node2vec, AdaDIF (λ , mode)
  - Extra labels needed by Planetoid / GCNN for early stopping

- HK and PR run to convergence -- AdaDIF relies just on *K=20*

Multiclass graphs

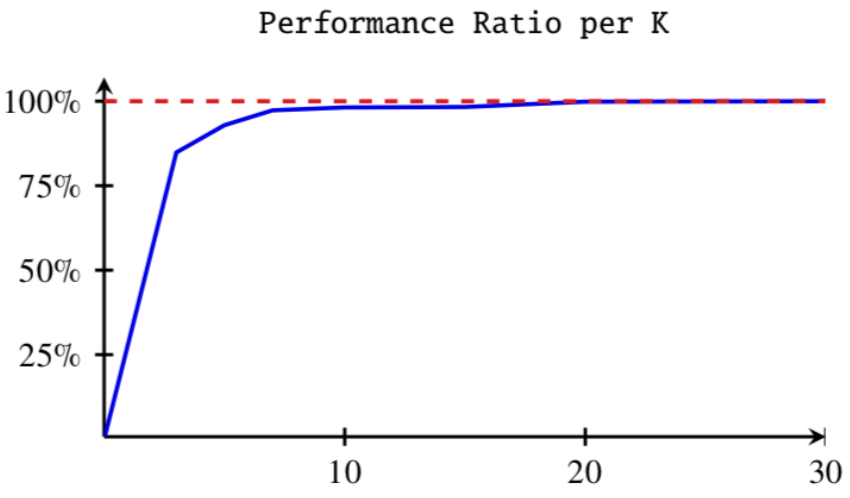
- State-of-the-art performance
  - Large margin improvement over Citeseer

	Graph	Cora			Citeseer			PubMed		
	$\|\mathcal{L}_c\|$	5	10	20	5	10	20	5	10	20
Micro-F1	AdaDIF	67.5 ± 2.2	$\mathbf{71.0 \pm 2.0}$	$\mathbf{73.2 \pm 1.2}$	$\mathbf{42.3 \pm 4.4}$	$\mathbf{49.5 \pm 3.0}$	$\mathbf{53.5 \pm 1.2}$	62.0 ± 6.0	68.5 ± 4.5	$\mathbf{74.1 \pm 1.7}$
	PPR	67.1 ± 2.3	70.2 ± 2.1	72.8 ± 1.5	41.1 ± 5.2	48.7 ± 2.5	52.5 ± 0.9	$\mathbf{63.1 \pm 1.1}$	$\mathbf{69.5 \pm 3.8}$	$\mathbf{74.1 \pm 1.8}$
	HK	67.0 ± 2.5	70.5 ± 2.5	72.9 ± 1.2	40.0 ± 5.6	48.0 ± 2.4	51.8 ± 1.1	62.0 ± 0.6	68.3 ± 4.7	$\mathbf{74.0 \pm 1.8}$
	LP	61.8 ± 3.5	66.3 ± 4.2	71.0 ± 2.7	40.7 ± 2.5	48.0 ± 3.7	51.9 ± 1.3	56.2 ± 11.0	68.0 ± 6.1	69.3 ± 2.4
	Node2vec	$\mathbf{68.9 \pm 1.9}$	70.2 ± 1.6	72.4 ± 1.2	39.2 ± 3.7	46.5 ± 2.4	51.0 ± 1.4	61.7 ± 13.0	66.4 ± 4.6	71.1 ± 2.4
	Deepwalk	68.4 ± 2.0	70.0 ± 1.6	72.0 ± 1.4	38.4 ± 3.9	45.5 ± 2.0	50.4 ± 1.5	61.5 ± 1.3	65.8 ± 5.0	70.5 ± 2.2
	Planetoid-G	63.5 ± 4.7	65.6 ± 2.7	69.0 ± 1.5	37.8 ± 4.0	44.9 ± 3.3	49.8 ± 1.4	60.7 ± 2.0	63.4 ± 2.3	68.0 ± 1.5
	GCN	60.1 ± 3.7	65.5 ± 2.5	68.6 ± 1.9	38.3 ± 3.2	44.2 ± 2.2	48.0 ± 1.8	60.0 ± 1.9	63.6 ± 2.5	70.5 ± 1.5
Macro-F1	AdaDIF	$\mathbf{65.5 \pm 2.5}$	$\mathbf{70.6 \pm 2.2}$	$\mathbf{72.0 \pm 1.1}$	$\mathbf{36.1 \pm 3.9}$	$\mathbf{44.0 \pm 2.8}$	$\mathbf{48.1 \pm 1.2}$	60.4 ± 0.6	67.0 ± 4.4	$\mathbf{72.6 \pm 1.8}$
	PPR	65.0 ± 2.3	70.0 ± 2.3	71.9 ± 1.5	34.7 ± 5.0	43.5 ± 2.3	47.6 ± 0.6	$\mathbf{61.7 \pm 0.6}$	$\mathbf{68.1 \pm 3.6}$	$\mathbf{72.6 \pm 1.8}$
	HK	65.0 ± 2.5	70.0 ± 2.6	$\mathbf{72.0 \pm 1.1}$	33.9 ± 5.4	42.8 ± 2.2	47.0 ± 0.6	60.5 ± 0.6	66.8 ± 4.7	$\mathbf{72.7 \pm 1.8}$
	LP	60.1 ± 3.2	66.5 ± 4.1	70.6 ± 2.3	34.8 ± 4.6	41.8 ± 3.9	51.5 ± 1.2	51.5 ± 12.3	66.2 ± 6.6	67.8 ± 2.0
	Node2vec	62.4 ± 2.0	64.7 ± 1.7	69.2 ± 1.2	34.6 ± 2.7	41.6 ± 1.9	45.3 ± 1.5	59.5 ± 1.2	64.0 ± 3.8	72.3 ± 1.4
	Deepwalk	61.8 ± 2.2	64.5 ± 2.0	68.5 ± 1.4	34.0 ± 2.5	41.0 ± 2.0	44.7 ± 1.8	59.3 ± 1.2	63.8 ± 4.0	72.1 ± 1.3
	Planetoid-G	59.9 ± 4.5	63.0 ± 3.0	68.7 ± 1.9	33.3 ± 2.5	40.2 ± 2.2	43.6 ± 2.0	57.7 ± 1.5	61.9 ± 3.5	66.1 ± 1.8
	GCN	53.8 ± 6.6	61.9 ± 2.6	63.8 ± 1.5	32.8 ± 2.0	39.1 ± 1.8	43.0 ± 1.7	54.4 ± 4.1	57.2 ± 5.2	60.5 ± 2.4

Experimental Results II

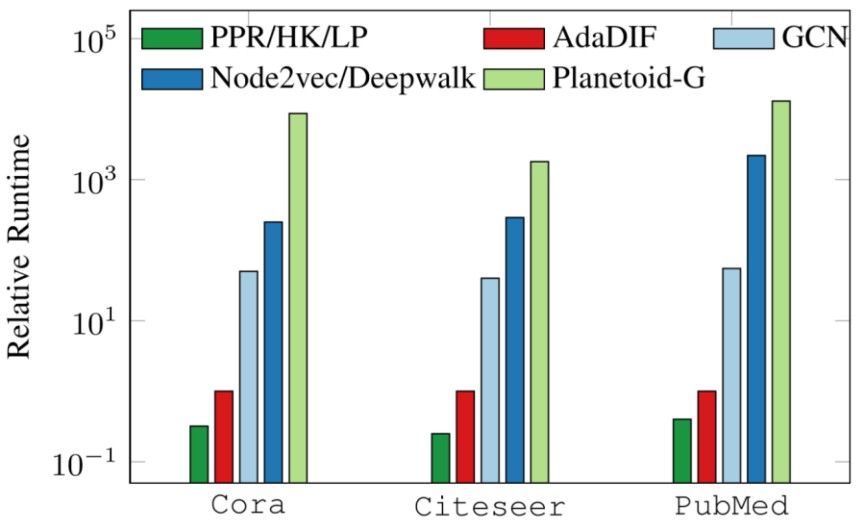
Effect of K

- Peak performance is typically achieved for K around 20



Runtime Comparisons

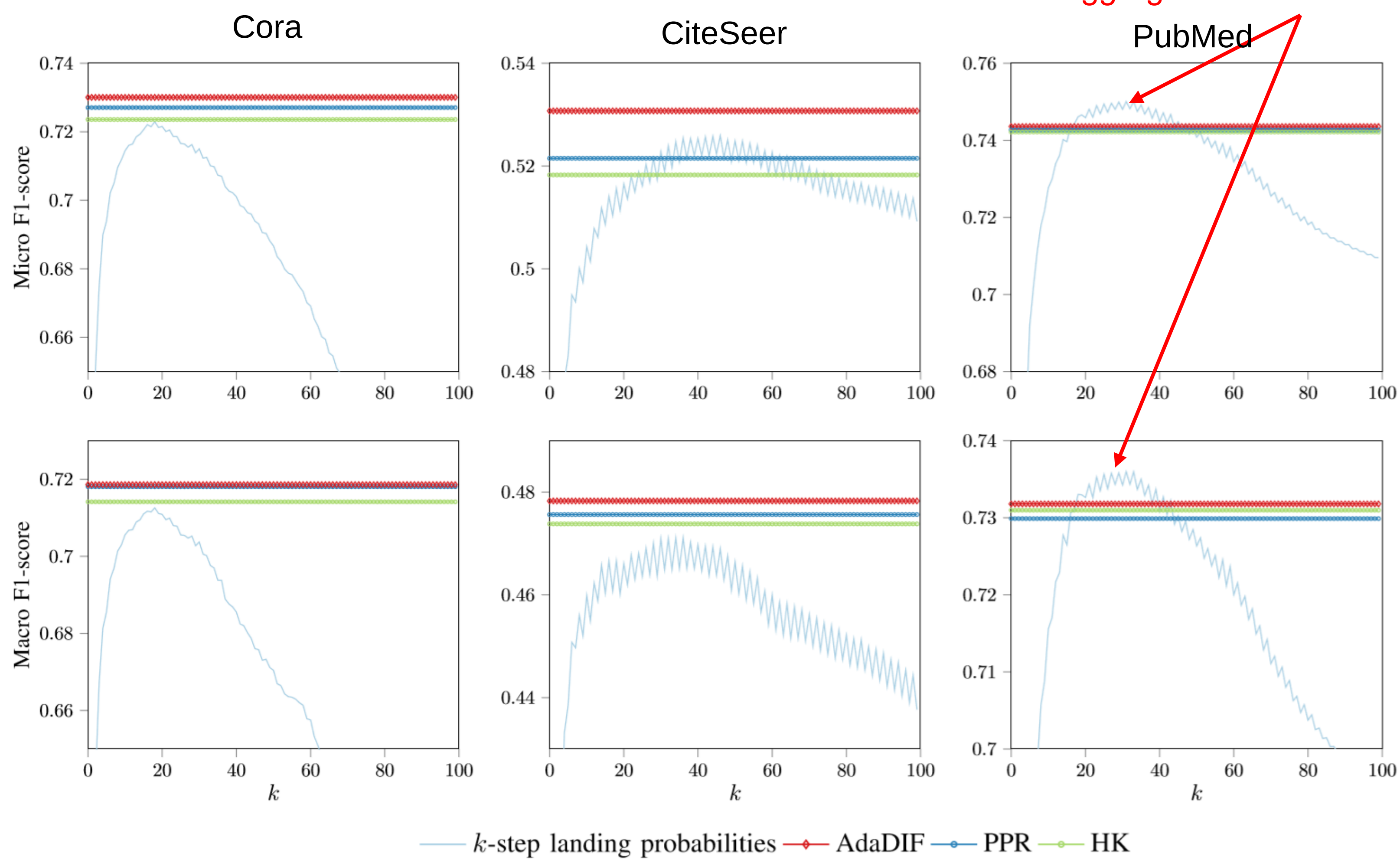
- AdaDIF is **significantly faster** than competing approaches



Per-step analysis

- Accuracy of k-th landing probabilities is a type of “graph-signature”

Aggregation doesn’t always help !



D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis, ”Adaptive Diffusions for Scalable Learning over Graphs”, IEEE Transactions on Signal Processing 2019 (short version received Best Paper Award in KDD MLG '18)

Multilabel graphs

- Number of labels per node assumed known (typical)
  - Evaluate accuracy of top-ranking classes
- AdaDIF approaches Node2vec Micro-F1 accuracy for PPI and BlogCatalog
  - Significant improvement over non-adaptive PPR and HK for all graphs
- AdaDIF achieves state-of-the-art Macro-F1 performance

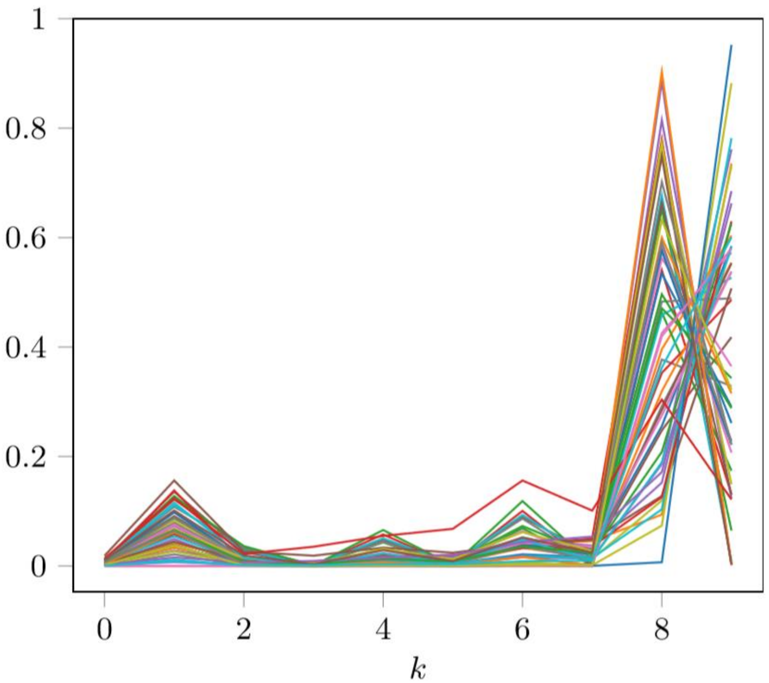
	Graph	PPI			BlogCatalog			Wikipedia		
	$\lvert\mathcal{L}\rvert/\lvert\mathcal{V}\rvert$	10%	20%	30%	10%	20%	30%	10%	20%	30%
Micro-F1	AdaDIF	15.4 ± 0.5	17.9 ± 0.7	$\mathbf{19.2 \pm 0.6}$	31.5 ± 0.6	34.4 ± 0.5	36.3 ± 0.4	28.2 ± 0.9	30.0 ± 0.5	31.2 ± 0.7
	PPR	13.8 ± 0.5	15.8 ± 0.6	17.0 ± 0.4	21.1 ± 0.8	23.6 ± 0.6	25.2 ± 0.6	10.5 ± 1.5	8.1 ± 0.7	7.2 ± 0.5
	HK	14.5 ± 0.5	16.7 ± 0.6	18.1 ± 0.5	22.2 ± 1.0	24.7 ± 0.7	26.6 ± 0.7	9.3 ± 1.4	7.3 ± 0.7	6.0 ± 0.7
	Node2vec	$\mathbf{16.5 \pm 0.6}$	$\mathbf{18.2 \pm 0.3}$	19.1 ± 0.3	$\mathbf{35.0 \pm 0.3}$	$\mathbf{36.3 \pm 0.3}$	$\mathbf{37.2 \pm 0.2}$	$\mathbf{42.3 \pm 0.9}$	$\mathbf{44.0 \pm 0.6}$	$\mathbf{45.1 \pm 0.4}$
	Deepwalk	16.0 ± 0.6	17.9 ± 0.5	18.8 ± 0.4	34.2 ± 0.4	35.7 ± 0.3	36.4 ± 0.4	41.0 ± 0.8	43.5 ± 0.5	44.1 ± 0.5
Macro-F1	AdaDIF	$\mathbf{13.4 \pm 0.6}$	$\mathbf{15.4 \pm 0.7}$	$\mathbf{16.5 \pm 0.7}$	$\mathbf{23.0 \pm 0.6}$	$\mathbf{25.3 \pm 0.4}$	$\mathbf{27.0 \pm 0.4}$	$\mathbf{7.7 \pm 0.3}$	$\mathbf{8.3 \pm 0.3}$	$\mathbf{9.0 \pm 0.2}$
	PPR	12.9 ± 0.4	14.7 ± 0.5	15.8 ± 0.4	17.3 ± 0.5	19.5 ± 0.4	20.8 ± 0.3	4.4 ± 0.3	3.8 ± 0.6	3.6 ± 0.2
	HK	$\mathbf{13.4 \pm 0.6}$	$\mathbf{15.4 \pm 0.5}$	$\mathbf{16.5 \pm 0.4}$	18.4 ± 0.6	20.7 ± 0.4	22.3 ± 0.4	4.2 ± 0.4	3.7 ± 0.5	3.5 ± 0.2
	Node2vec	13.1 ± 0.6	15.2 ± 0.5	16.0 ± 0.5	16.8 ± 0.5	19.0 ± 0.3	20.1 ± 0.4	7.6 ± 0.3	8.2 ± 0.3	8.5 ± 0.3
	Deepwalk	12.7 ± 0.7	15.1 ± 0.6	16.0 ± 0.5	16.6 ± 0.5	18.7 ± 0.5	19.6 ± 0.4	7.3 ± 0.3	8.1 ± 0.2	8.2 ± 0.2

Diversity of class diffusions

Q: Why does AdaDIF perform much better than fixed HK/PPR in m. label case **?**

A: Possibly due to large number of classes with diverse distributions….
AdaDIF naturally captures this diversity.

Plot of different class diffusion parameters for a 10% sample of BlogCatalog



https://github.com/DimBer/SSL_lib

Anomaly identification - removal

- **Leave-one-out loss**: Quantifies how well each node is predicted by the rest

$$\ell^c_{\text{rob}}(\mathbf{y}_{\mathcal{L}_c}, \boldsymbol{\theta}) := \sum_{i \in \mathcal{L}} \frac{1}{d_i} \left([\bar{\mathbf{y}}_{\mathcal{L}_c}]_i - [\mathbf{f}_c(\boldsymbol{\theta}; \mathcal{L} \setminus i)]_i \right)^2$$

- $\mathbf{f}_c(\boldsymbol{\theta}; \mathcal{L} \setminus i)$'s obtained via $|\mathcal{L}|$ different random walks ($\mathcal{O}(|\mathcal{L}|K|\mathcal{E}|)$)

- Model outliers as large residuals, captured by nnz entries of sparse vec. $\mathbf{o} \in \mathbb{R}^N$

$$\ell^c_{\text{rob}}(\mathbf{y}_{\mathcal{L}_c}, \mathbf{o}, \boldsymbol{\theta}) := \|\mathbf{D}_{\mathcal{L}}^{-\frac{1}{2}} \left(\mathbf{o} + \bar{\mathbf{y}}_{\mathcal{L}_c} - \mathbf{R}_c^{(K)} \boldsymbol{\theta} \right) \|_2^2$$

- Joint optimization

$$\{\hat{\boldsymbol{\theta}}_c, \hat{\mathbf{o}}_c\}_{c \in \mathcal{Y}} = \arg \min_{\substack{\boldsymbol{\theta}_c \in \mathcal{S}^K \\ \mathbf{o}_c \in \mathbb{R}^N}} \sum_{c \in \mathcal{Y}} \left[\ell^c_{\text{rob}}(\mathbf{y}_{\mathcal{L}_c}, \mathbf{o}_c, \boldsymbol{\theta}_c) + \lambda_\theta \|\boldsymbol{\theta}_c\|_2^2 \right] + \lambda_o \|\mathbf{D}_{\mathcal{L}}^{-\frac{1}{2}} \mathbf{O}\|_{2,1}$$

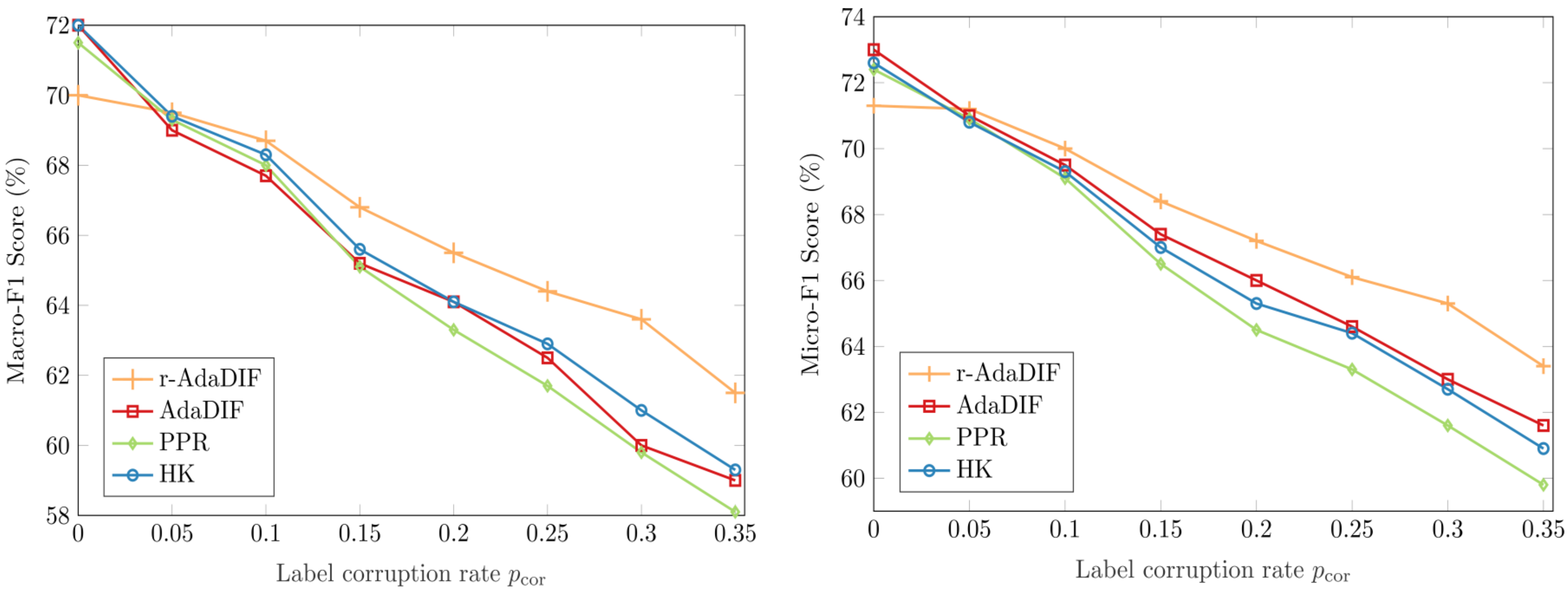
Group sparsity on $\mathbf{O} := [\mathbf{o}_1 \cdots \mathbf{o}_{|\mathcal{Y}|}]$
i.e., force consensus among classes regarding which nodes are outliers

- Alternating minimization converges to stationary point

- Remove outliers $\mathcal{S} := \{i \in \mathcal{L} : \|[\hat{\mathbf{O}}]_{i,:}\|_2 > 0\}$ from $\mathcal{L}$ and predict $\mathcal{U}$ using $\{\hat{\boldsymbol{\theta}}_c\}_{c \in \mathcal{Y}}$

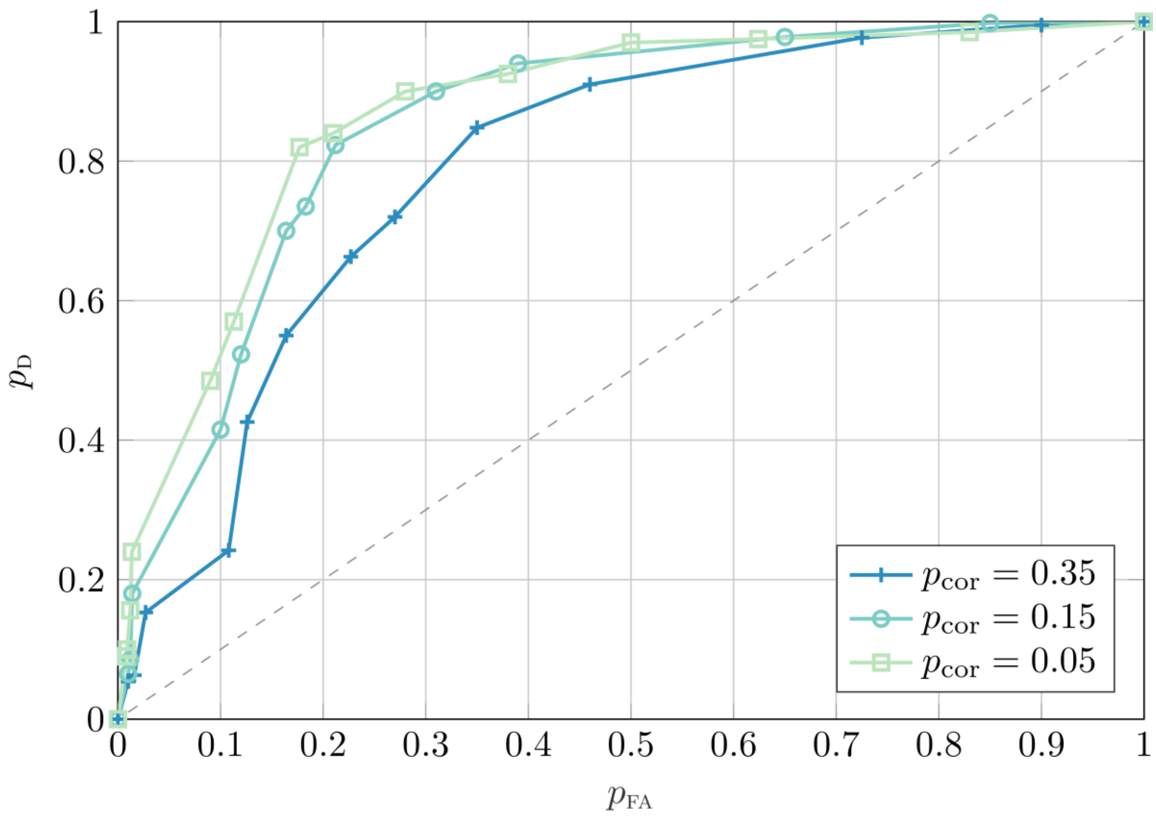
Testing classifier robustness

- Anomalies injected in Cora graph
  - Go through each entry $[\mathbf{y}_{\mathcal{L}}]_i = c$ of $\mathbf{y}_{\mathcal{L}}$
  - With probability p_{cor} draw a label $c' \sim \text{Unif}\{\mathcal{Y} \setminus c\}$
  - Replace $[\mathbf{y}_{\mathcal{L}}]_i \leftarrow c'$
- For fixed $\lambda_o > 0$, accuracy with $p_{\text{cor}} > 0$ improves as false samples are removed
  - Less accuracy for $p_{\text{cor}} = 0$ (no anomalies), only useful samples removed (false alarms)

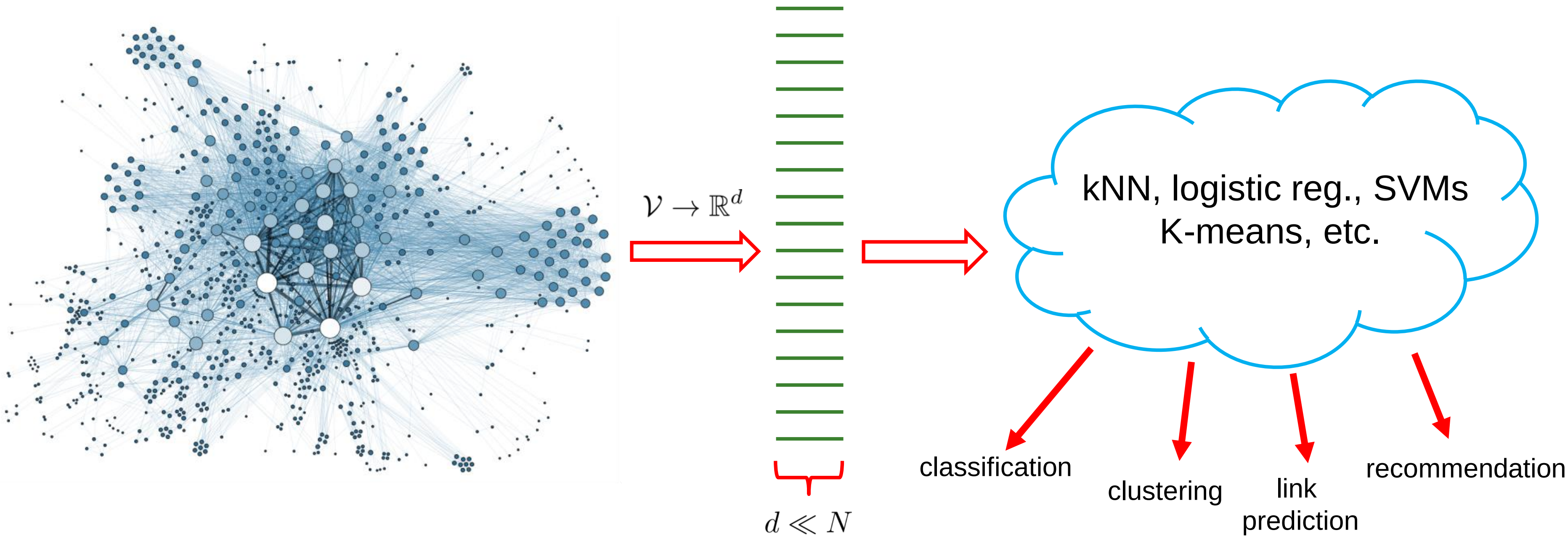


Testing anomaly detection performance

- **ROC** curve: Probability of detection vs probability of false alarms
  - As expected, performance improves as p_{cor} decreases



Unsupervised node embedding



Objective: Per-node feature extraction preserving graph **structure** and **properties**

- Aim to preserve **some** pairwise similarity $s_{\mathcal{G}}(\cdot,\cdot) : \mathcal{V} \times \mathcal{V} \to \mathcal{R}$

$$\{\mathbf{e}_i^*\}_{i\in\mathcal{V}} = \arg \min_{\{\mathbf{e}_i\}_{i\in\mathcal{V}}} \sum_{i,j\in\mathcal{V}} \ell\left(\boxed{s_{\mathcal{G}}(v_i, v_j)}, s_{\mathcal{E}}(\mathbf{e}_i, \mathbf{e}_j)\right)$$

critical

H. Cai, V. W. Zheng, and K. Chang, “A comprehensive survey of graph embedding: problems, techniques and applications,” IEEE Trans. on Knowledge and Data Engineering, vol. 30, no. 9, pp. 1616– 1637, 2018.

Node Embedding via matrix factorization

- For loss $\ell(x, x') = (x - x')^2$ and $\mathbb{R}^d$ similarities $s_{\mathcal{E}}(\mathbf{e}_i, \mathbf{e}_j) = \mathbf{e}_i^T \mathbf{e}_j$

- Embedding ≡ Low-rank factorization of (symmetric) $\mathbf{S}_{\mathcal{G}}$

$$\mathbf{E}^* = \arg \min_{\mathbf{E} \in \mathbb{R}^{N \times d}} \|\mathbf{S}_{\mathcal{G}} - \mathbf{E}\mathbf{E}^T\|_F^2$$

- Using Truncated(T) SVD $\mathbf{S}_{\mathcal{G}} := \mathbf{U}_d \mathbf{\Sigma}_d \mathbf{U}_d^T$ is $\boxed{\mathbf{E}^* = \mathbf{U}_d \sqrt{\mathbf{\Sigma}_d}}$
  - Fast if $\mathrm{nnz}(\mathbf{S}_{\mathcal{G}}) \ll N^2$ and $d \ll N$

- Most approaches use a fixed $\mathbf{S}_{\mathcal{G}}$
  - Few parametrize $\mathbf{S}_{\mathcal{G}}$ and tune parameters using labels (e.g., Nod2vec)

Our contribution: Adapt $\mathbf{S}_{\mathcal{G}}$ to $\mathcal{G}$ efficiently and w/o supervision

Multi-length node similarities

- "Base" similarity $\mathbf{S}$ must follow graph sparsity pattern (e.g., $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$)

$$S_{i,j} = \begin{cases} s_{i,j}, & (i,j) \in \mathcal{E} \\ 0, & (i,j) \notin \mathcal{E} \end{cases}$$

- Similarity matrix parametrization
  - Weigh k-length (non-Hamiltonian) paths with θ_k

$$\mathbf{S}_{\mathcal{G}}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k \mathbf{S}^k, \quad \text{s.t.} \quad \boldsymbol{\theta} \in \mathcal{S}^K$$

- No explicit formation of dense $\mathbf{S}_{\mathcal{G}}(\boldsymbol{\theta})$
  - Only TSVD of $\mathbf{S}$ is needed
  - Polynomial obeyed by TSVD if $\theta_k \geq 0 \ \forall k$

$$\mathbf{S}_{\mathcal{G}}(\boldsymbol{\theta}) = \mathbf{U}\left(\sum_{k=1}^{K} \theta_k \boldsymbol{\Sigma}^k\right)\mathbf{U}^T \Longrightarrow \mathbf{E}^*(\boldsymbol{\theta}) = \mathbf{U}_d\sqrt{\boldsymbol{\Sigma}_d(\boldsymbol{\theta})}$$

Capturing spectral information

- If base similarity matrix is PSD

$$\mathbf{S} \in \mathcal{P}_N^+ \implies \mathrm{SVD}(\mathbf{S}) \equiv \mathrm{EVD}(\mathbf{S})$$

- Multi-length embeddings given as weighted **eigenvectors**

$$\mathbf{E}^*(\boldsymbol{\theta}) = \mathbf{U}_d \sqrt{\boldsymbol{\Lambda}_d(\boldsymbol{\theta})}$$

- All requirements (symmetry, sparsity pattern, PSD) can be met

$$\boxed{\mathbf{S} = \frac{1}{2}\left(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)}$$

  - Can be shown that $\lambda_i\left(\mathbf{S}^k\right) \in [0,1] \ \forall \ i, k$
  - Same eigenvectors as spectral clustering

$$\mathbf{L}_{\mathrm{sym}} := \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

  - Large weights to longer paths shrink “detailed” eigenvectors

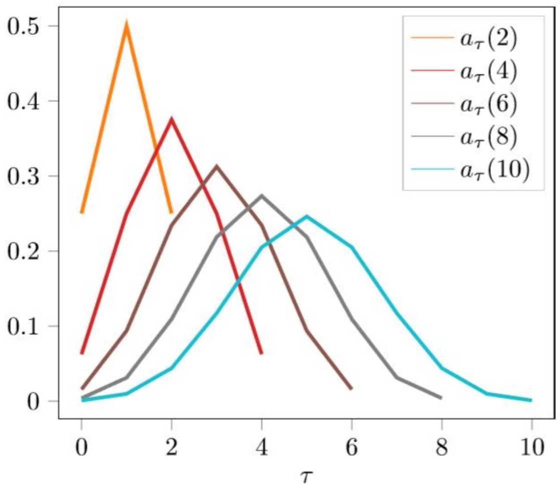
Random-walk interpretation

- Node similarity as function of landing probabilities weighted at different lengths
  - Each length is **not** freely parametrized (lazy random walks)
  - Dictionary-of-diffusions type

$$s_{\mathcal{G}}(v_i, v_j, \boldsymbol{\theta}) = \sqrt{\frac{d_j}{d_i}} \sum_{\tau=0}^{K} c_\tau(\boldsymbol{\theta}) \Pr\{X_\tau = v_i | X_0 = v_j\}$$

$$c_\tau(\boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k \alpha_\tau(k)$$

$$\alpha_\tau(k) = \begin{cases} \frac{1}{2^k}\binom{k}{\tau}, & 0 \le \tau \le k \\ 0, & \text{else} \end{cases} \rightarrow$$



Numerical study of model

- Assume edges are generated according to model

$$\mathbf{A} \sim f_A(\mathbf{A})$$

- "True" similarities

$$s^*(v_i, v_j) := \Pr\{(i,j) \in \mathcal{E}\} = \mathbb{E}_{f_A}\left[A_{i,j}\right]$$

$$\mathbf{S}^* := \mathbb{E}_{f_A}\left[\mathbf{A}\right]$$

- Quality-of-match (QoM) of estimated similarities $\hat{\mathbf{S}} = F(\mathbf{A})$

$$\mathrm{QoM} := \mathbb{E}_{f_A}\left[\mathrm{PC}\left(\mathbf{S}^*, F(\mathbf{A})\right)\right]$$

$$\mathrm{PC}\left(\mathbf{X}_1, \mathbf{X}_2\right) := \frac{\left(\mathrm{vec}\left(\mathbf{X}_1\right)\right)^T \mathrm{vec}\left(\mathbf{X}_2\right)}{\|\mathbf{X}_1\|_F \|\mathbf{X}_2\|_F}$$

Numerical experiments on SBMs

- Stochastic block model with 3 clusters of equal size
- SBM probabilities matrix (p>q, c<1)

$$\mathbf{W}_{\mathrm{sbm}} = \begin{bmatrix} p & q & cq \\ q & p & q \\ cq & q & p \end{bmatrix}$$

- "True" similarities given by SBM parameters

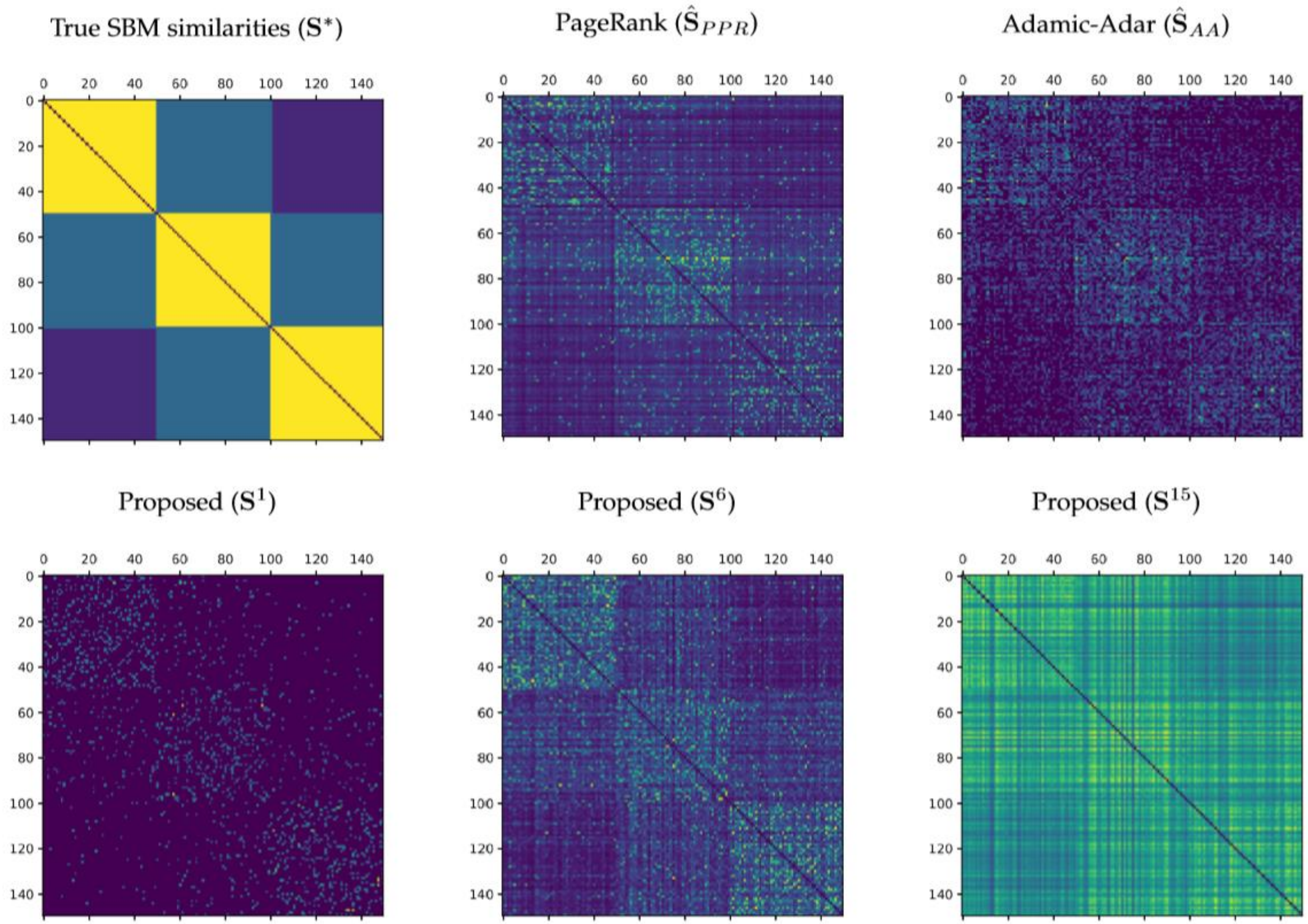
$$\mathbf{S}^* = \mathbb{E}[\mathbf{A}] = \mathbf{W}_{\mathrm{sbm}} \otimes \left(\mathbf{1}_{N/3}\mathbf{1}_{N/3}^T\right) - \mathrm{diag}(p\mathbf{1}_N)$$

- Evaluation of different scenarios with N=150, and 100 experiments
  - Comparison of $\mathbf{S}^k$ with baseline node similarities

$$\hat{\mathbf{S}}_{PPR} := (1-\alpha)(\mathbf{I} - \alpha\mathbf{A}\mathbf{D}^{-1})^{-1} \qquad \hat{\mathbf{S}}_{NEIGH} := \mathbf{A}^2$$

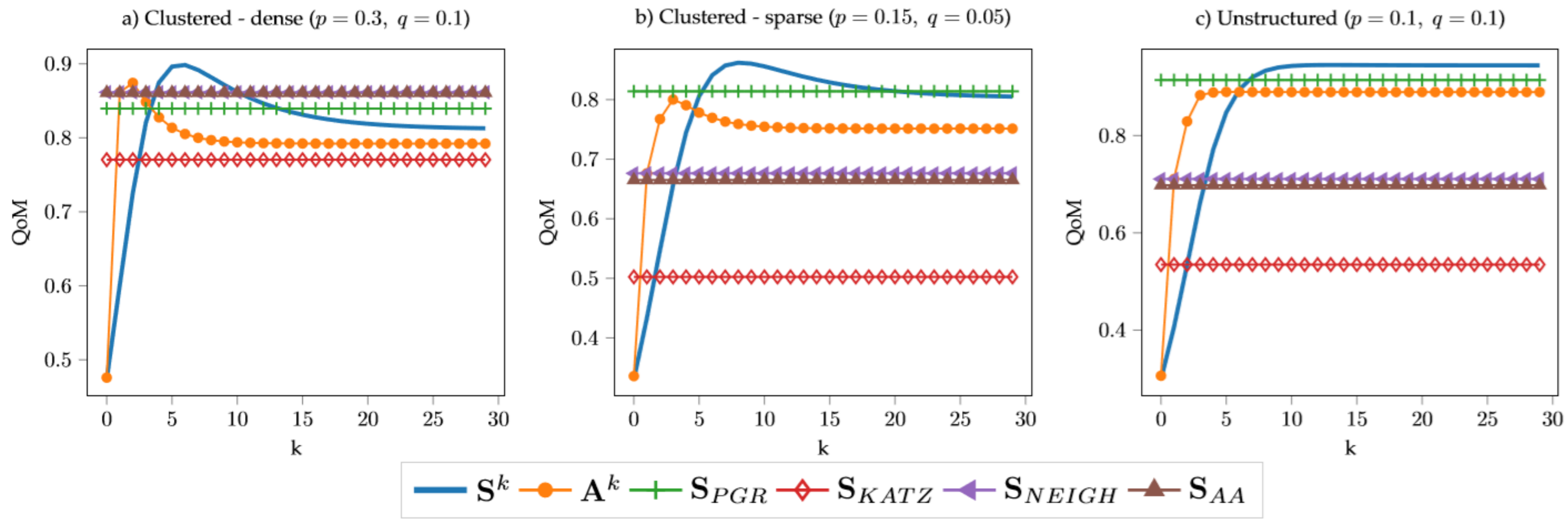
$$\hat{\mathbf{S}}_{KATZ} := (1-\beta)(\mathbf{I} - \beta\mathbf{A})^{-1}\mathbf{A} \qquad \hat{\mathbf{S}}_{AA} := \mathbf{A}\mathbf{D}^{-1}\mathbf{A}$$

Behavior of various similarities



https://github.com/DimBer/ASE-project/tree/master/sim_tests

Quality of match (QoM) results



Disclaimer: To be determined whether $\mathbf{S}^k$ can yield superior link prediction

- Main observations
  - For structured graphs there exists a "sweet spot" of k's
  - $\mathbf{S}^k$ can match "true" similarities better than $\mathbf{A}^k$

Q: Can we find the "sweet spot" from only one $\mathbf{A}$**?**

D. Berberidis and G. B. Giannakis, " Adaptive-similarity node embedding for Scalable Learning over Graphs", IEEE Transactions on Knowledge and Data Engineering (submitted 2018)

Adaptive Similarity Embedding (ASE)

Step 1) Draw edge samples $\mathcal{S}^+ \subset \mathcal{E}$ and $\mathcal{S}^- \subset \mathcal{V} \times \mathcal{V} \setminus \mathcal{E}$ with $|\mathcal{S}^+| = |\mathcal{S}^-|$

- Samples must be <u>representative</u> but w. <u>min. spectral perturbation</u>*
- Sampling wp $\propto \dfrac{d_i + d_j}{d_i d_j}$ very simple & strikes a good balance

Step 2) Build $\mathcal{G}^- = \left(\mathcal{V}, \mathcal{E} \setminus \mathcal{S}^+\right)$ and do TSVD on $\mathbf{S}^-$

- Convenient embedding similarity parametrization $\left(\mathbf{e}_i^-(\boldsymbol{\theta})\right)^T \mathbf{e}_j^-(\boldsymbol{\theta}) = \mathbf{x}_{i,j}^T \ \boldsymbol{\theta}$

Step 3) Train SVM parameters $\boldsymbol{\theta}^*$ to separate $\mathcal{S}^+$ and $\mathcal{S}^-$

- Use $\mathbf{x}_{i,j}$'s for $(i,j) \in \mathcal{S}^+ \cup \mathcal{S}^-$ as features

Step 4) Repeat Steps 1-3 for different splits if variance is large (small sample)

Step 5) TSVD on $\mathbf{S}$ of full $\mathcal{G}$ and return $\mathbf{E}^*(\boldsymbol{\theta}^*) = \mathbf{U}_d \sqrt{\boldsymbol{\Sigma}_d(\boldsymbol{\theta}^*)}$

* A. Milanese, J. Sun, and T. Nishikawa, "Approximating spectral impact of structural perturbations in large networks," Physical Review E, vol. 81, no. 4, pp. 046–112, 2010.

Experiments on real graphs

Competing baselines

- DeepWalk [Perozzi et al, '14]
- VERSE [Tsitsulin et al, '18]
- LINE [Tang et al, '15]
- HOPE [Ou et al, '16]
- Spectral (unweighted)

Graph	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{Y}\|$	Density
PPI (H. Sapiens)	3,890	76,584	50	10^{-2}
Wikipedia	4,733	184,182	40	1.6×10^{-2}
BlogCatalog	10,312	333,983	39	6.2×10^{-3}
ca-CondMat	23,133	93,497	-	3.5×10^{-4}
ca-AstroPh	18,772	198,110	-	1.1×10^{-3}
email-Enron	36,692	183,831	-	2.7×10^{-4}
CoCit	44,312	195,362	15	2×10^{-4}
vk2016-17	78,593	2,680,542	-	8.7×10^{-4}
com-Amazon	334,863	925,872	-	1.7×10^{-5}
com-DBLP	317,080	1,049,866	-	2.1×10^{-5}

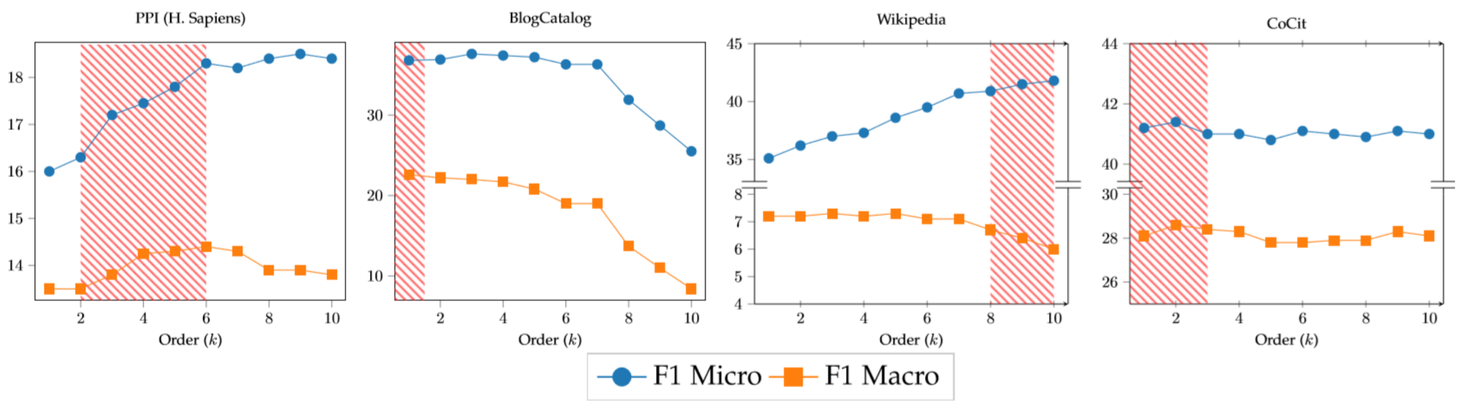
- Comparison with
  - Scalable methods $\cancel{\mathcal{O}(N^3)}$ $\cancel{\mathcal{O}(N^2)}$
  - No (or standardized) hyper-parameters
- Embedding dimension d = 100 (typical) for all methods
- ASE maximum length K=10 (since typically $\theta_k = 0$ for k >10)
- Embeddings used as features for classification, link-prediction, and clustering

Validating parameter adaptation with labels

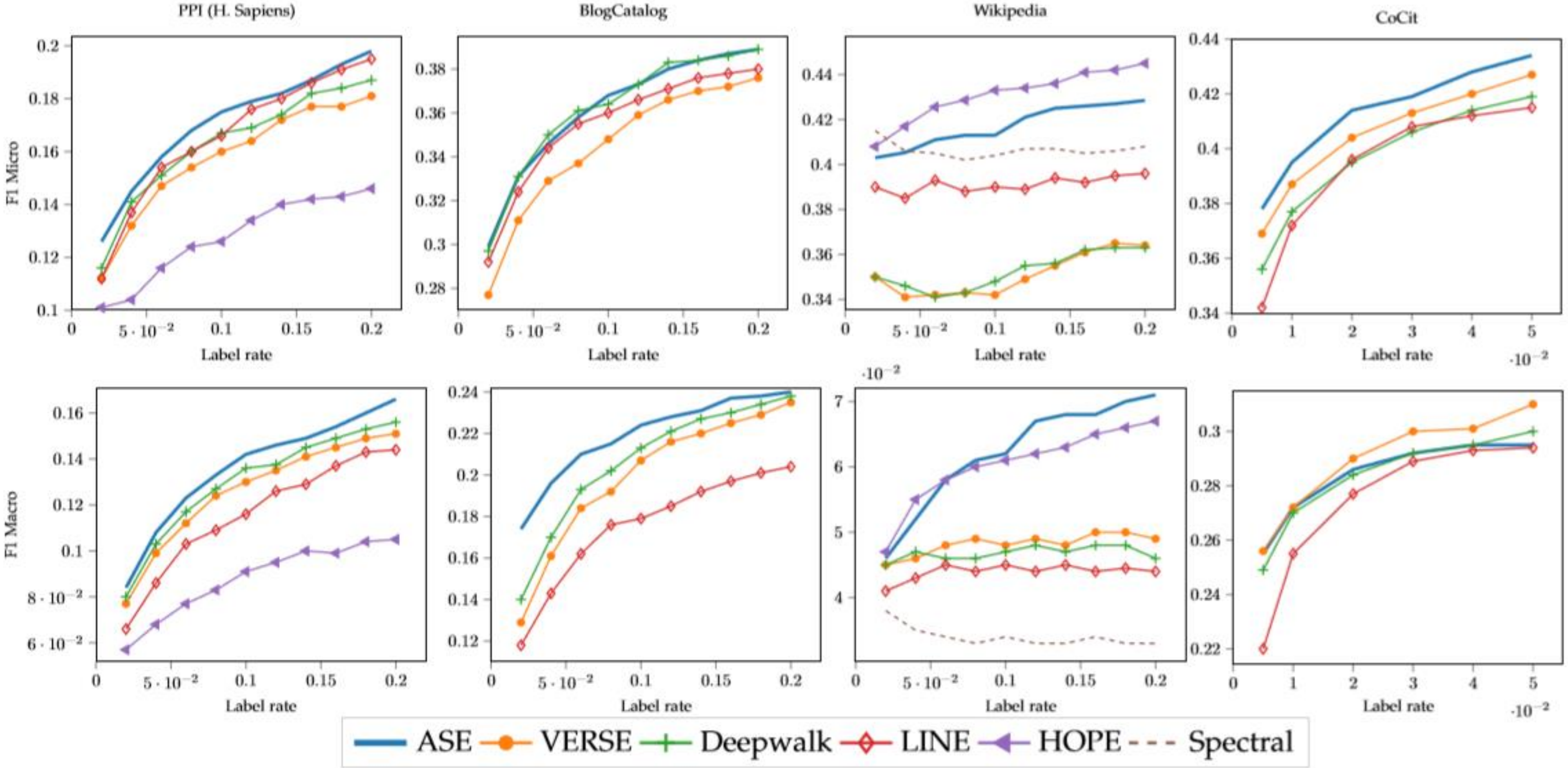
- Variability of ASE parameters among graphs

Graph	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	range	strength
PPI (H. Sapiens)	0.00	**0.14**	**0.31**	**0.29**	**0.21**	**0.04**	0.00	0.00	0.00	0.00	medium	medium
Wikipedia	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.01**	**0.37**	**0.62**	long	strong
BlogCatalog	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	short	very strong
ca-CondMat	**0.55**	**0.33**	**0.12**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	short	strong
ca-AstroPh	**0.76**	**0.24**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	short	strong
email-Enron	**0.24**	**0.25**	**0.18**	**0.14**	**0.1**	**0.06**	**0.02**	0.00	0.00	0.00	medium	weak
CoCit	**0.61**	**0.33**	**0.06**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	short	strong
vk2016-17	**0.71**	**0.29**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	short	strong
com-Amazon	**0.10**	**0.10**	**0.10**	**0.10**	**0.09**	**0.09**	**0.09**	**0.09**	**0.09**	**0.09**	short	very weak
com-DBLP	**0.11**	**0.10**	**0.10**	**0.09**	**0.09**	**0.09**	**0.09**	**0.09**	**0.09**	**0.08**	short	very weak

- ASE parameters >0 for lengths that perform well on labels
  - Fully Unsupervised: No cross-validation or a-priori knowledge of labels



Node classification with logistic regression



- ASE has the highest accuracy in 5/8 cases
  - Not clear which method is second best
  - Spectral (unweighted) embeddings perform poorly

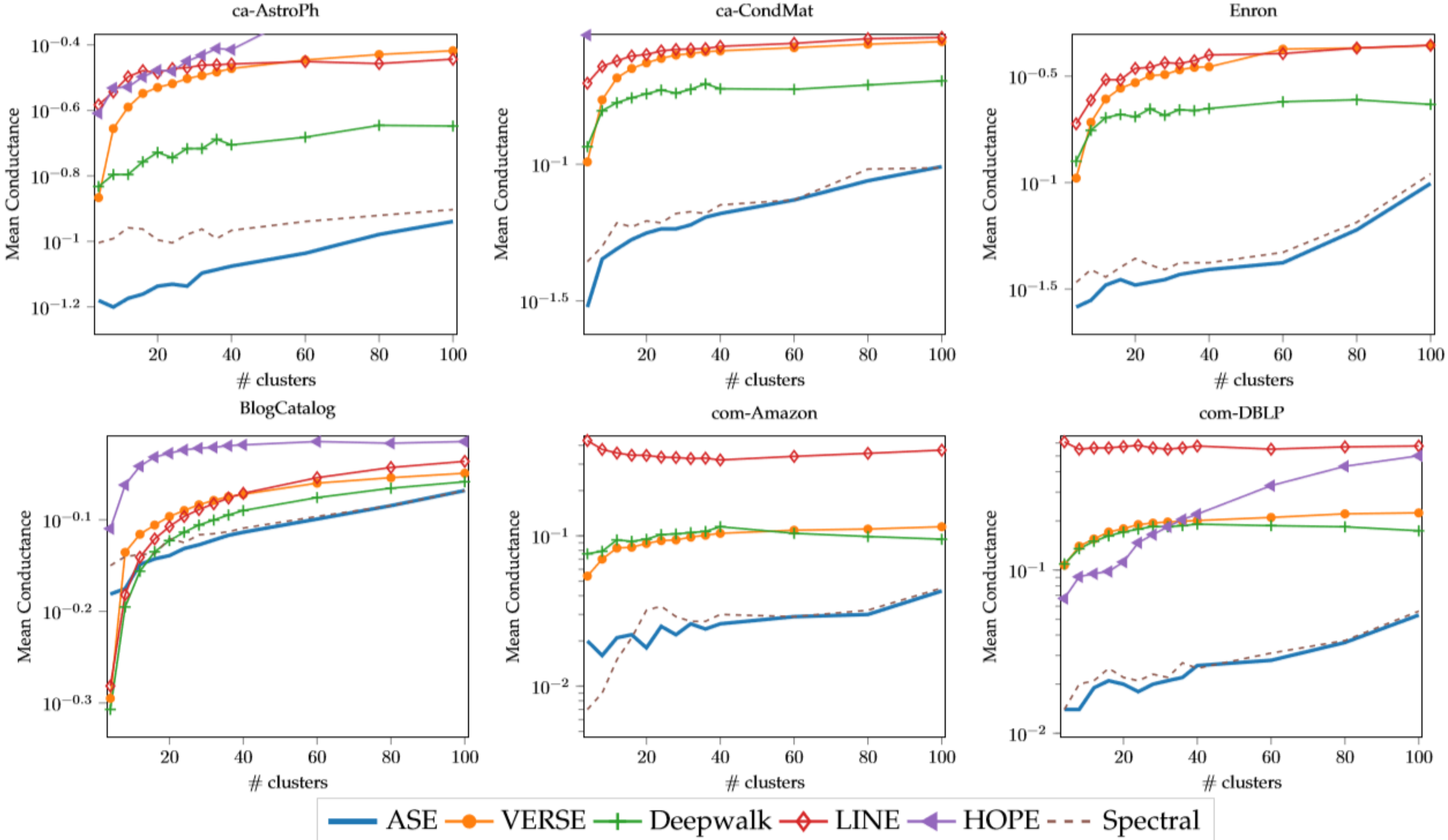
Link prediction on VK social network

- New friendships (≈ 20,000) appeared between Nov. 2016 and May 2017
  - Only Nov. 2016 users considered
- Experiment [Tsitsulin et al., ‘18]
  - Embeded Nov. 2016 network
  - Sample ≈ 20,000 ``negative’’ edges
  - Split positive and negative new edges to 50/50 training/testing
  - Train logistic regression using Nov. 2016 features (on training edges)
  - Classify test edges to positive and negative

	VERSE	ASE	LINE	Deepwalk	HOPE	Spectral
Acc.	0.79	0.75	0.74	0.69	0.62	0.60

- ASE second best
  - Much more accurate than unweighted spectral embedding

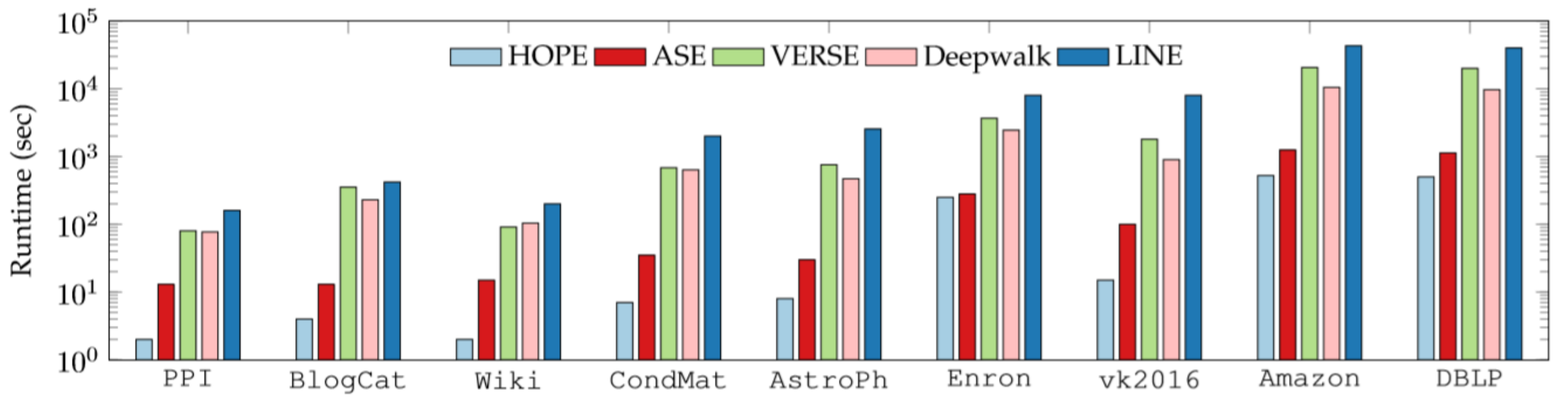
Clustering with K-means++



- Evaluating **average conductance per cluster** wrt # of clusters
- ASE “inherits” spectral clustering properties (high resolution limit)

Runtime

- SVD based methods (ASE and HOPE) are very fast!



- Results are for shared-memory multi-threaded setup
  - SLEPc with MPI (although for shared memory) was used for SVD
  - SVD more memory demanding than LINE & VERSE
  - LINE & VERSE could benefit more from massive parallelization

https://github.com/DimBer/ASE-project/tree/master/portable
https://github.com/DimBer/ASE-project/tree/master/scalable

Conclusions

- Diffusion / Random Walk – based approaches
  - Simple, intuitive and flexible tool for graph - learning tasks
    - Semi-supervised: Node classification
    - Unsupervised: Node Embedding
  - Scalable to large graphs
- Observations
  - Semi-supervised
    - Simple models capture most of the information in “simple” data
    - Adaptation to graph/class can boost performance in more complex cases
  - Unsupervised
    - Each graph has unique diffusion-based similarity pattern
    - Such similarities can be identified with relative accuracy

Related work and Ongoing Projects

- Personalized Diffusions for Top-N recommendation
  - Random walks on (inferred) item graphs
  - Adapting random-walk pattern of each user based on history
- Robust Semi-Supervised Classification
  - RANdom Sampling And Consensus (RANSAC) + Diffusion-based classifiers
- Binary Node Embeddings / Node Hashing
  - Each node is mapped to *d* bits
  - Suitable for large networks (> 1 million nodes)
  - Aim to compress graph and facilitate learning/mining tasks (e.g., kNN queries)

A. N. Nikolakopoulos, D. Berberidis, G. Karypis, and G. B. Giannakis, "Personalized Diffusions for Top-N Recommendation," International Conference on Machine Learning, submitted 2019.

Thank you !









Leave-one-out fitting loss

- Quantifies how well each (labeled) node is predicted by the rest

$$\ell^c_{\text{rob}}(\mathbf{y}_{\mathcal{L}_c}, \boldsymbol{\theta}) := \sum_{i \in \mathcal{L}} \frac{1}{d_i} \left([\bar{\mathbf{y}}_{\mathcal{L}_c}]_i - [\mathbf{f}_c(\boldsymbol{\theta}; \mathcal{L} \setminus i)]_i \right)^2$$

- $\mathbf{f}_c(\boldsymbol{\theta}; \mathcal{L} \setminus i)$'s obtained via $|\mathcal{L}|$ different random walks ($\mathcal{O}(|\mathcal{L}|K|\mathcal{E}|)$)

$$\mathbf{f}_c(\boldsymbol{\theta}; \mathcal{L} \setminus i) = \begin{cases} \mathbf{f}_c(\boldsymbol{\theta}), & i \notin \mathcal{L}_c \\ \mathbf{f}_c(\boldsymbol{\theta}; \mathcal{L}_c \setminus i), & i \in \mathcal{L}_c \end{cases} \quad \mathbf{f}_c(\boldsymbol{\theta}; \mathcal{L}_c \setminus i) = \sum_{k=1}^{K} \theta_k \mathbf{p}^{(k)}_{\mathcal{L}_c \setminus i}$$

$$\mathbf{p}^{(k)}_{\mathcal{L}_c \setminus i} := \mathbf{H}^k \mathbf{v}_{\mathcal{L}_c \setminus i} \qquad [\mathbf{v}_{\mathcal{L}_c \setminus i}]_j = \begin{cases} 1/|\mathcal{L}_c \setminus i|, & j \in \mathcal{L}_c \setminus i \\ 0, & \text{else} \end{cases}$$

- Compact form

$$\ell^c_{\text{rob}}(\mathbf{y}_{\mathcal{L}_c}, \boldsymbol{\theta}) := \|\mathbf{D}_{\mathcal{L}}^{-\frac{1}{2}} \left(\bar{\mathbf{y}}_{\mathcal{L}_c} - \mathbf{R}^{(K)}_c \boldsymbol{\theta} \right) \|_2^2 \qquad \left[\mathbf{R}^{(K)}_c\right]_{ik} := \begin{cases} \left[\mathbf{p}^{(k)}_{\mathcal{L}_c \setminus i}\right]_i, & i \in \mathcal{L}_c \\ \left[\mathbf{p}^{(k)}_c\right]_i, & \text{else} \end{cases}$$

- Diffusion parameters

$$\hat{\boldsymbol{\theta}}_c = \arg \min_{\boldsymbol{\theta} \in \mathcal{S}^K} \ell^c_{\text{rob}}(\mathbf{y}_{\mathcal{L}_c}, \boldsymbol{\theta}) + \lambda_\theta \|\boldsymbol{\theta}\|_2^2$$

Anomaly identification - removal

- Model outliers as large residuals, captured by nnz entries of sparse vec. $\mathbf{o} \in \mathbb{R}^N$

$$\ell^c_{\text{rob}}(\mathbf{y}_{\mathcal{L}_c}, \mathbf{o}, \boldsymbol{\theta}) := \|\mathbf{D}_{\mathcal{L}}^{-\frac{1}{2}} \left(\mathbf{o} + \bar{\mathbf{y}}_{\mathcal{L}_c} - \mathbf{R}_c^{(K)}\boldsymbol{\theta}\right)\|_2^2$$

- Joint optimization

$$\{\hat{\boldsymbol{\theta}}_c, \hat{\mathbf{o}}_c\}_{c\in\mathcal{Y}} = \arg \min_{\substack{\boldsymbol{\theta}_c \in \mathcal{S}^K \\ \mathbf{o}_c \in \mathbb{R}^N}} \sum_{c\in\mathcal{Y}} \left[\ell^c_{\text{rob}}(\mathbf{y}_{\mathcal{L}_c}, \mathbf{o}_c, \boldsymbol{\theta}_c) + \lambda_\theta \|\boldsymbol{\theta}_c\|_2^2\right] + \lambda_o \|\mathbf{D}_{\mathcal{L}}^{-\frac{1}{2}}\mathbf{O}\|_{2,1}$$

Group sparsity on

$\mathbf{O} := [\mathbf{o}_1 \cdots \mathbf{o}_{|\mathcal{Y}|}]$

i.e., force consensus among classes regarding which nodes are outliers

- While, $\|\hat{\boldsymbol{\theta}}_c^{(t)} - \hat{\boldsymbol{\theta}}_c^{(t-1)}\|_\infty \leq \epsilon, \ \forall c \in \mathcal{Y}$ iterate:

$$\hat{\boldsymbol{\theta}}_c^{(t)} = \arg \min_{\boldsymbol{\theta}\in\mathcal{S}^K} \ell^c_{\text{rob}}(\bar{\mathbf{y}}_{\mathcal{L}_c} + \hat{\mathbf{o}}_c^{(t-1)}, \boldsymbol{\theta}) + \lambda_\theta \|\boldsymbol{\theta}\|_2^2$$

$$\hat{\mathbf{O}}^{(t)} = \text{SoftThres}_{\lambda_o}\left(\tilde{\mathbf{Y}}^{(t)}\right)$$

Residuals

$$\tilde{\mathbf{Y}}^{(t)} := \left[\tilde{\mathbf{y}}_1^{(t)}, \ldots, \mathbf{y}_{|\mathcal{Y}|}^{(t)}\right]$$

$$\tilde{\mathbf{y}}_c^{(t)} := \bar{\mathbf{y}}_{\mathcal{L}_c} - \mathbf{R}_c^{(K)}\hat{\boldsymbol{\theta}}_c^{(t)}$$

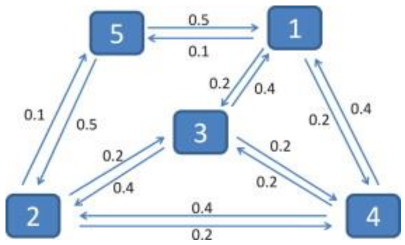
Row-wise soft-thresholding

$$\mathbf{Z} = \text{SoftThres}_{\lambda_o}(\mathbf{X})$$

$$\mathbf{z}_i = \|\mathbf{x}_i\|_2[1 - \lambda_o/(2\|\mathbf{x}_i\|_2)]_+$$

- Alternating minimization converges to stationary point
- Remove outliers $\mathcal{S} := \{i \in \mathcal{L} : \|[\hat{\mathbf{O}}]_{i,:}\|_2 > 0\}$ from $\mathcal{L}$ and predict $\mathcal{U}$ using $\{\hat{\boldsymbol{\theta}}_c\}_{c\in\mathcal{Y}}$

Random walks on graphs



- Position of random walker at step *k* : $X_k \in \mathcal{V}$
  - Transition probabilities

$$\begin{aligned} \Pr\{X_k = i | X_{k-1} = j\} &= W_{ij}/d_j \\ &:= [\mathbf{H}]_{ij} = [\mathbf{W}\mathbf{D}^{-1}]_{ij} \end{aligned}$$

- Steady-state probs.

$$\pi_i := \lim_{k\to\infty} \sum_{j\in\mathcal{V}} \Pr\{X_k = i | X_0 = j\} \Pr\{X_0 = j\} = \frac{d_i}{2|\mathcal{E}|}$$

  - Presumes undirected, connected, and non-bipartite graphs
  - **Not** informative for SSL

- Step-*k* landing probabilities

$$p_i^{(k)} := \sum_{j\in\mathcal{V}} \Pr\{X_k = i | X_0 = j\} \Pr\{X_0 = j\}$$

$$\mathbf{p}^{(k)} = \mathbf{H}^k \mathbf{p}^{(0)} := [p_1^{(k)} \dots p_N^{(k)}]^T$$

  - Measure influence of $\mathbf{p}^{(0)}$ on every node in $\mathcal{V}$ - informative for SSL!

Landing probabilities for SSL

- Random walk per class with $\mathbf{p}_c^{(k)} = \mathbf{H}^k \mathbf{v}_c$ $\quad \mathbf{P}_c^{(K)} := \left[\mathbf{p}_c^{(1)} \quad \cdots \quad \mathbf{p}_c^{(K)}\right]$
  - Initial ("root") probability distribution $\quad [\mathbf{v}_c]_i = \begin{cases} 1/|\mathcal{L}_c|, & i \in \mathcal{L}_c \\ 0, & \text{else} \end{cases}$
  - Per step landing probabilities found by multiplying with sparse ***H*** $\quad \mathcal{L}_c := \{i \in \mathcal{L} : y_i = c\}$
- Family of per-class *diffusions*

$$\mathbf{f}_c(\boldsymbol{\theta}) := \sum_{k=1}^{K} \theta_k \mathbf{p}_c^{(k)} = \mathbf{P}_c^{(K)} \boldsymbol{\theta}, \quad \boldsymbol{\theta} \in \mathcal{S}^K$$

  - Valid pmf with ***K*-dim** probability simplex

$$\mathcal{S}^K := \{\boldsymbol{\theta} \in \mathbb{R}^K : \ \boldsymbol{\theta} \geq \mathbf{0}, \ \mathbf{1}^\top \boldsymbol{\theta} = 1\}$$

- Max-likelihood per-node classifier

$$\hat{y}_i(\boldsymbol{\theta}) := \arg\max_{c \in \mathcal{Y}} \left[\mathbf{f}_c(\boldsymbol{\theta})\right]_i$$

Unifying diffusion-based SSL

Special case 1: Personalized page rank (PPR) diffusion [Lin'10]

$$\mathbf{f}_c(\boldsymbol{\theta}_{\text{PPR}}) = (1-\alpha)\sum_{k=1}^{K} \alpha^k \mathbf{p}_c^{(k)} \qquad \boldsymbol{\theta}_{\text{PPR}} := (1-\alpha)\left[\alpha \cdots \alpha^K\right]^{\top}, \ \alpha \in (0,1)$$

- Pmf of random walk with restart probability 1-α ; in steady-state

$$\lim_{K\to\infty} \mathbf{f}_c(\boldsymbol{\theta}_{\text{PPR}}) = (1-\alpha)(\mathbf{I}-\alpha\mathbf{H})^{-1}\mathbf{v}_c$$

Special case 2: Heat kernel (HK) diffusion [Chung'07]

$$\mathbf{f}_c(\boldsymbol{\theta}_{\text{HK}}) = e^{-t}\sum_{k=0}^{K} \frac{t^k}{k!}\mathbf{p}_c^{(k)} \qquad \boldsymbol{\theta}_{\text{HK}} := e^{-t}\left[t \quad \frac{t^2}{2} \quad \cdots \quad \frac{t^K}{K!}\right]^{\top}, \ t > 0$$

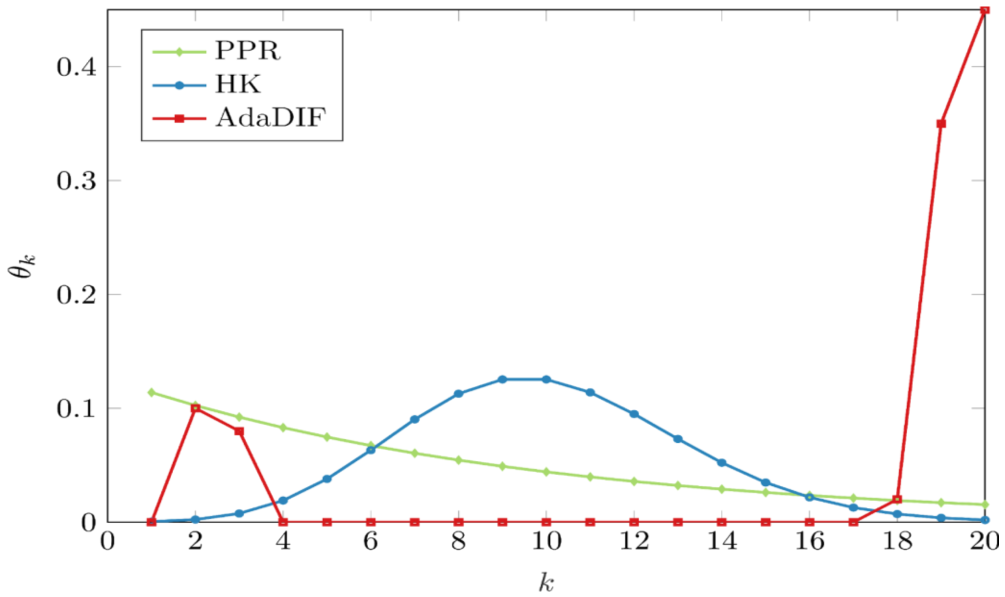
- "Heat" flowing from roots after time *t* ; in steady-state

$$\lim_{K\to\infty} \mathbf{f}_c(\boldsymbol{\theta}_{\text{HK}}) = e^{-t(\mathbf{I}-\mathbf{H})}\mathbf{v}_c$$

- HK and PPR have fixed parameters (t, α)

Our key contribution: Graph- and label-adaptive selection of $\boldsymbol{\theta}_c \in \mathcal{S}^K$

Interpretation



- For $\lambda \to \infty$ (smoothness-only), $\hat{\boldsymbol{\theta}}_c \to \mathbf{e}_K$
  - Weights concentrates on last landing prob.
- For $\lambda \to 0$ (fit-only)
  - Weights concentrate on first few landing prob.
- The simplex constrain promotes **sparsity** in the diffusion coefficients

D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis, "AdaDIF: Adaptive Diffusions for Efficient Semi-supervised Learning over Graphs", Proc. of IEEE Intl. Conf. on Big Data, Seattle, WA, Dec. 2018.

Adaptive diffusions

$$\hat{\mathbf{f}}_c = \arg\min_{\mathbf{f}\in\mathbb{R}^N} \ell(\mathbf{y}_{\mathcal{L}_c}, \mathbf{f}) + \lambda R(\mathbf{f})$$

Normalized label indicator vector

$$\ell(\mathbf{y}_{\mathcal{L}_c}, \mathbf{f}) = \sum_{i\in\mathcal{L}} \frac{1}{d_i}(y_i - f_i)^2 = (\bar{\mathbf{y}}_{\mathcal{L}_c} - \mathbf{f})^\top \mathbf{D}_{\mathcal{L}}^{-1} (\bar{\mathbf{y}}_{\mathcal{L}_c} - \mathbf{f})$$

$$R(\mathbf{f}) = \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left(\frac{f_i}{d_i} - \frac{f_j}{d_j}\right)^2 = \mathbf{f}^\top \mathbf{D}^{-1}\mathbf{L}\mathbf{D}^{-1}\mathbf{f}$$

- **AdaDIF** scalable to large-scale graphs (*K* << *N*)

$$\hat{\boldsymbol{\theta}}_c = \arg\min_{\boldsymbol{\theta}\in\mathcal{S}^K} \ell(\mathbf{y}_{\mathcal{L}_c}, \mathbf{f}_c(\boldsymbol{\theta})) + \lambda R(\mathbf{f}_c(\boldsymbol{\theta}))$$

- Linear-quadratic $\hat{\boldsymbol{\theta}}_c = \arg\min_{\boldsymbol{\theta}\in\mathcal{S}^K} \boldsymbol{\theta}^\top \mathbf{A}_c \boldsymbol{\theta} + \boldsymbol{\theta}^\top \mathbf{b}_c$

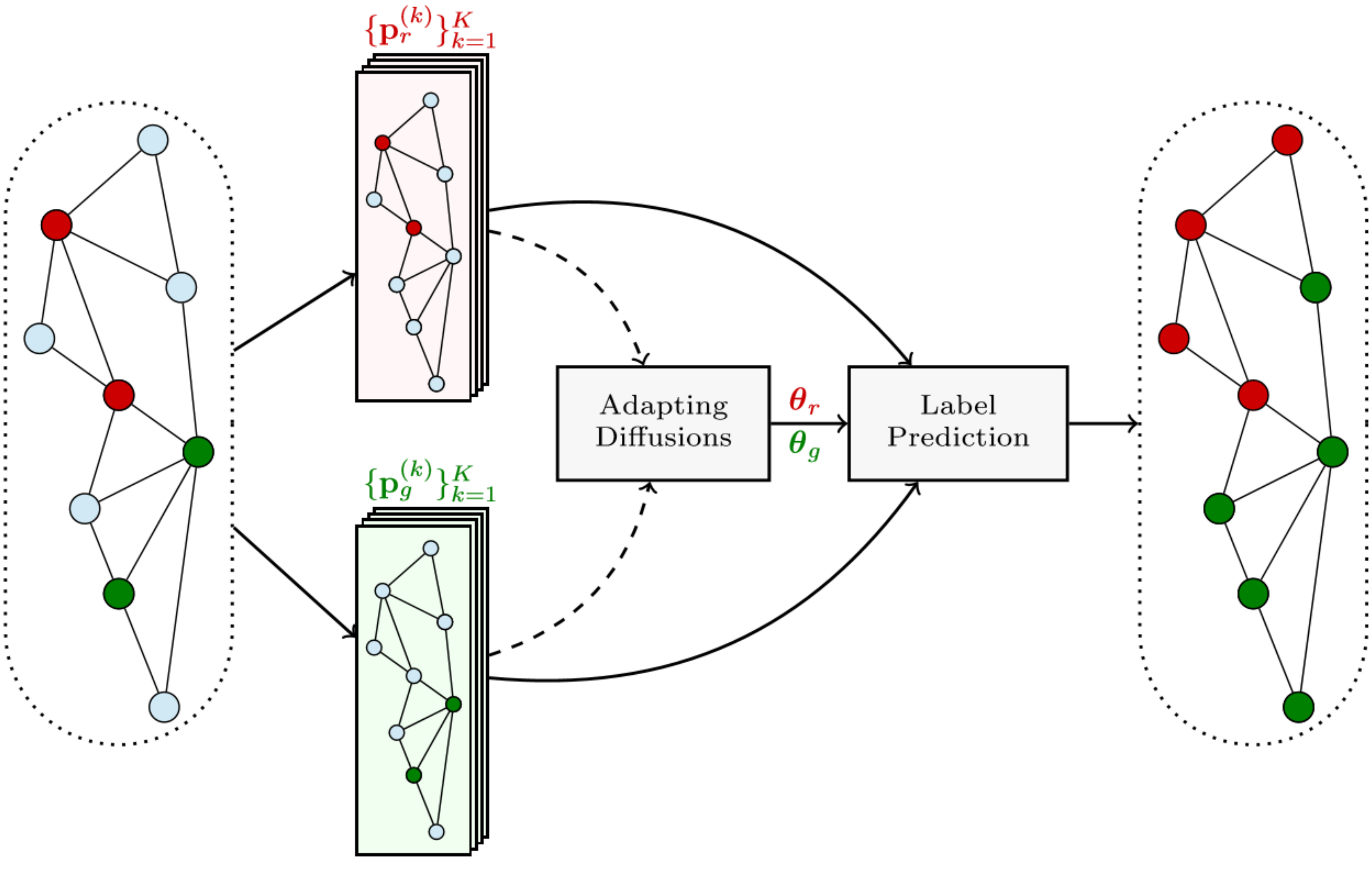
$$\mathrm{b}_c = -\frac{2}{|\mathcal{L}|}(\mathrm{P}_c^{(K)})^\top \mathrm{D}_{\mathcal{L}}^{-1}\mathrm{y}_{\mathcal{L}^c}$$

$$\mathbf{A}_c = (\mathbf{P}_c^{(K)})^\top \left(\mathbf{D}_{\mathcal{L}}^{-1}\mathbf{P}_c^{(K)} + \lambda \mathbf{D}^{-1}\tilde{\mathbf{P}}_c^{(K)}\right)$$

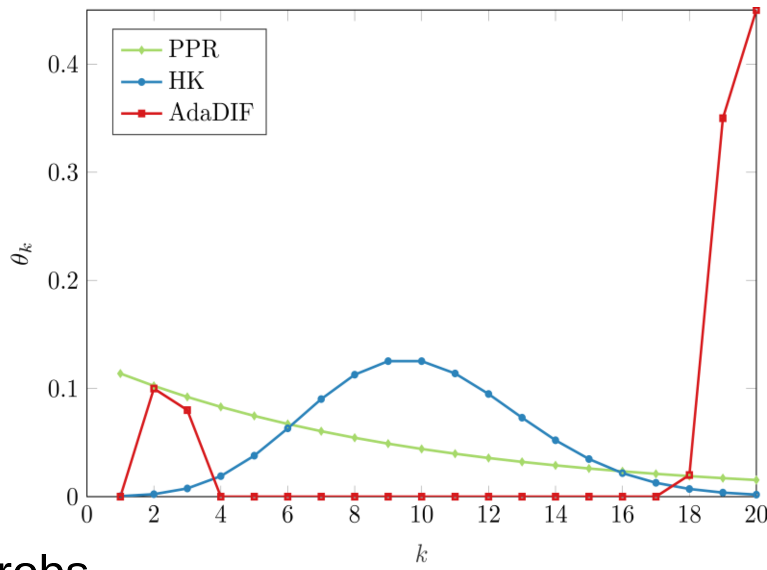
``Differential'' landing prob.

$$\tilde{\mathbf{p}}_c^{(k)} := \mathbf{p}_c^{(k)} - \mathbf{p}_c^{(k+1)}$$

AdaDIF in a nutshell



Interpretation and complexity



$$\hat{\boldsymbol{\theta}}_c = \arg\min_{\boldsymbol{\theta}\in\mathcal{S}^K} \ell(\mathbf{y}_{\mathcal{L}_c}, \mathbf{f}_c(\boldsymbol{\theta})) + \lambda R(\mathbf{f}_c(\boldsymbol{\theta}))$$

- For $\lambda \to \infty$ (smoothness-only), $\hat{\boldsymbol{\theta}}_c \to \mathbf{e}_K$
  - Weight concentrates on last landing prob.
- For $\lambda \to 0$ (fit-only)
  - Weight concentrates on first few landing probs
  - **Intuition:** very short walks visit similarly labeled nodes
- AdaDIF targets a "sweet-spot" between the two
  - Simplex constraint promotes sparsity on $\boldsymbol{\theta}$
- If $K < |\mathcal{E}|/N$, per-class complexity $\mathcal{O}(|\mathcal{E}|K)$ thanks to sparsity of ***H***
  - Same as non-adaptive HK and PPR; also parallelizable across classes
  - Reflect on PPR and Google … just avoid *K* >>

On the choice of *K*

Definition. Let $\mathbf{p}_+$ and $\mathbf{p}_-$ denote respectively the seed vectors for nodes of class "+" and "-," initializing the landing probability vectors in matrices $\mathbf{X}_c := \mathbf{P}_c^{(K)}$, and $\check{\mathbf{X}}_c := \left[\mathbf{p}_c^{(1)} \cdots \mathbf{p}_c^{(K-1)} \mathbf{p}_c^{(K+1)}\right]$, $c \in \{+,-\}$. With $\mathbf{y} := \mathbf{X}_+\boldsymbol{\theta} - \mathbf{X}_-\boldsymbol{\theta}$ and $\check{\mathbf{y}} := \check{\mathbf{X}}_+\boldsymbol{\theta} - \check{\mathbf{X}}_-\boldsymbol{\theta}$ the γ-distinguishability threshold of the diffusion-based classifier is the smallest integer K_γ satisfying $\|\mathbf{y} - \check{\mathbf{y}}\| \leq \gamma$.

Theorem. For any diffusion-based classifier with coefficients $\boldsymbol{\theta}$ constrained to a probability simplex of appropriate dimensions, it holds that

$$K_\gamma \leq \frac{1}{\mu'} \log\left[\frac{2\sqrt{d_{\max}}}{\gamma}\left(\sqrt{\frac{1}{d_{\min_-}|\mathcal{L}_-|}} + \sqrt{\frac{1}{d_{\min_+}|\mathcal{L}_+|}}\right)\right]$$

$d_{\min+} := \min_{i\in\mathcal{L}_+} d_i,\ d_{\min-} := \min_{j\in\mathcal{L}_-} d_j,\ d_{\max} := \max_{i\in\mathcal{V}} d_i$ and $\mu' := \min\{\mu_2, 2-\mu_N\}$, $\{\mu_n\}_{n=1}^N$ eigenvalues of the normalized graph Laplacian in ascending order.

- Message: Increasing *K* does not help distinguishing between classes
  - Large *K* may even degrade performance due to over-parametrization

Unsupervised similarity learning

Algorithm 1 ADAPTIVE SIMILARITY EMBEDDING

Input: $\mathcal{G}$ **Output:** $\mathbf{E}$

// Training phase
$\boldsymbol{\Theta} = \emptyset$
while $|\boldsymbol{\Theta}| < T_s$ **do**
  $\mathcal{G}^-, \mathcal{S}^+, \mathcal{S}^- = \text{SAMPLE EDGES}(\mathcal{G})$
  $\boldsymbol{\theta}^*_{\mathcal{S}} = \text{TRAIN PARAMETERS}(\mathcal{G}^-, \mathcal{S}^+, \mathcal{S}^-)$
  $\boldsymbol{\Theta} = \boldsymbol{\Theta} \cup \boldsymbol{\theta}^*_{\mathcal{S}}$
end while
$\boldsymbol{\theta}^* = T_s^{-1} \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \boldsymbol{\theta}$

// Embedding phase
$\mathbf{S} = \frac{1}{2}\left(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)$
$\mathbf{S} = \mathbf{U}_d \boldsymbol{\Sigma}_d \mathbf{U}_d^T$
$\boldsymbol{\Sigma}_d(\boldsymbol{\theta}^*) = \sum_{k=1}^{K} \theta^*_k \boldsymbol{\Sigma}_d^k$
return $\mathbf{E} = \mathbf{U}_d \sqrt{\boldsymbol{\Sigma}_d(\boldsymbol{\theta}^*)}$

Algorithm 4 SIMPLEXSVM

Input: $\mathcal{X}, \mathcal{S}^+, \mathcal{S}^-$ **Output:** $\boldsymbol{\theta}^*$
$\boldsymbol{\theta}_0 = \frac{1}{K}\mathbf{1},\ t = 1$
while $\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\|_\infty \geq \text{tol}$ **do**
  $t = t + 1,\ \eta_t = a/\sqrt{t}$
  $\mathcal{S}_a^+ = \{e \in \mathcal{S}^+ \mid \mathbf{x}_e^T \boldsymbol{\theta}_{t-1} \leq \epsilon\}$
  $\mathcal{S}_a^- = \{e \in \mathcal{S}^- \mid \mathbf{x}_e^T \boldsymbol{\theta}_{t-1} \geq -\epsilon\}$
  $\mathbf{g}_t = \sum_{e \in \mathcal{S}_a^-} \mathbf{x}_e - \sum_{e \in \mathcal{S}_a^+} \mathbf{x}_e$
  $\mathbf{z}_t = (1 - 2\eta_t\lambda)\boldsymbol{\theta}_{t-1} - \frac{\eta_t}{N_s}\mathbf{g}_t$
  $\boldsymbol{\theta}_t = \text{SIMPLEXPROJ}(\mathbf{z}_t)$
end while
return $\boldsymbol{\theta}_t$

Algorithm 2 SAMPLE EDGES

Input: $\mathcal{G}$ **Output:** $\mathcal{G}^-, \mathcal{S}^+, \mathcal{S}^-$

// Sample edges
$\mathcal{S}^+ = \emptyset, \mathcal{G}^- = \mathcal{G}$
while $|\mathcal{S}^+| < N_s/2$ **do**
  Sample $v_1 \sim \text{Unif}(\mathcal{V})$
  if $|\mathcal{N}_{\mathcal{G}^-}(v_1)| > 1$ **then**
    Sample $v_2 \sim \text{Unif}(\mathcal{N}_{\mathcal{G}^-}(v_1))$
    if $|\mathcal{N}_{\mathcal{G}^-}(v_2)| > 1$ **then**
      $\mathcal{S}^+ = \mathcal{S}^+ \cup (v_1, v_2)$
      $\mathcal{G}^- = \mathcal{G}^- \setminus (v_1, v_2)$
    end if
  end if
end while

// Sample non-edges
$\mathcal{S}^- = \emptyset$
while $|\mathcal{S}^-| < N_s/2$ **do**
  Sample $(v_1, v_2) \sim \text{Unif}(\mathcal{V} \times \mathcal{V})$
  if $(v_1, v_2) \notin \mathcal{E}$ **then**
    $\mathcal{S}^- = \mathcal{S}^- \cup (v_1, v_2)$
  end if
end while
return $\mathcal{G}^-, \mathcal{S}^+, \mathcal{S}^-$

Algorithm 3 TRAIN PARAMETERS

Input: $\mathcal{G}, \mathcal{S}^+, \mathcal{S}^-$ **Output:** $\boldsymbol{\theta}^*_{\mathcal{S}}$
$\mathbf{S} = \frac{1}{2}\left(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)$
$\mathbf{S} = \mathbf{U}_d \boldsymbol{\Sigma}_d \mathbf{U}_d^T$
$\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$
Form $\mathcal{X}_{\mathcal{S}} = \{\mathbf{x}_{(i,j)}\}_{(i,j) \in \mathcal{S}}$ as in (30)
return $\boldsymbol{\theta}^*_{\mathcal{S}} = \text{SIMPLEXSVM}(\mathcal{X}_{\mathcal{S}}, \mathcal{S}^+, \mathcal{S}^-)$

ASE parameter sensitivity