One World Signal Processing Seminar Series

Adaptive Diffusions for Scalable and Robust Learning over Graphs

Georgios B. Giannakis

D. K. Berberidis

A. N. Nikolakopoulos

Acknowledgment: NSF 1711471 and 1901134

UNIVERSITY OF MINNESOTA
Driven to Discover℠

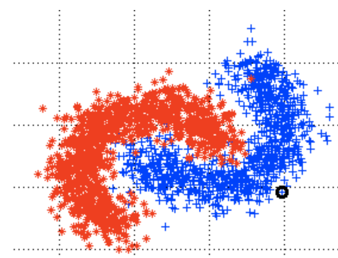
October 22, 2020

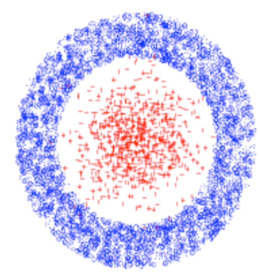
Motivation

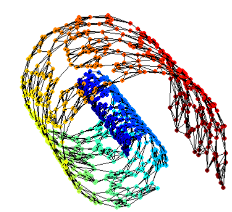
Graph representations

Real networks

Networks built on similarities







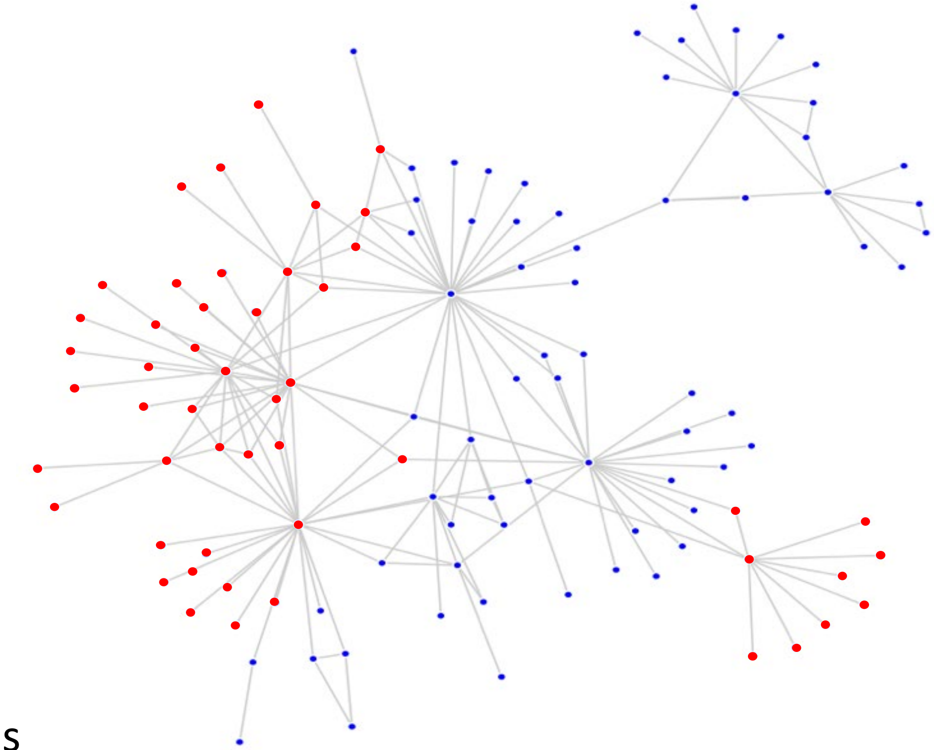
Objective: Learn values or labels of nodes from a subset; as e.g., in citation networks

Challenges: Graphs can be **huge**, and nodes are **scarcely labeled**

- Due to privacy, cost of battery, (un) reliable human annotators …

E. D. Kolaczyk, *Statistical Analysis of Network Data*, Springer, 2009.

Problem statement



- Graph $\mathcal{G} := \{\mathcal{V}, \mathcal{E}\}$
  - Weighted adjacency matrix ***W***
  - Label $y_i \in \mathcal{Y}$ per node v_i

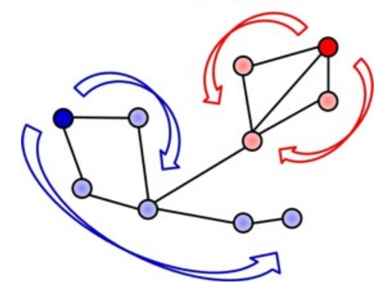
- Topology given or identifiable
  - Given in e.g. WSNs and social nets
  - Identifiable via e.g., nodal similarities

Goal: Given labels in $\mathcal{L} \subseteq \mathcal{V}$ and $\mathcal{G}$, infer unlabeled nodes in $\mathcal{U} := \mathcal{V} \setminus \mathcal{L}$

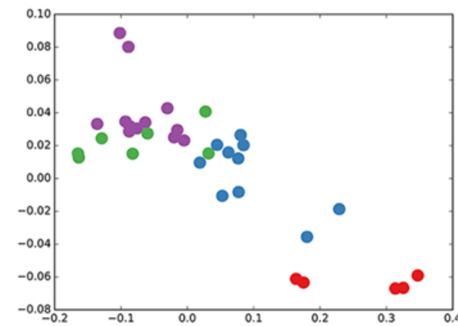
G. B. Giannakis, Y. Shen, and G. V. Karanikolas "Topology Identification and Learning over Graphs: Accounting for Nonlinearities and Dynamics," *Proceedings of the IEEE,* vol. 106, pp. 787-807, May 2018.

Work in context

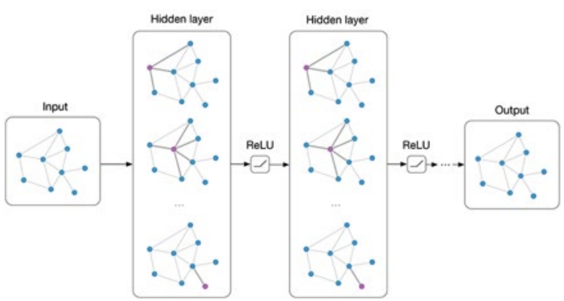
- Transductive semi-supervised learning (SSL) on graphs
  - Graph partitioning [Joachims et al‘03]
  - Manifold regularization [Belkin et al‘06]
  - Label propagation [Zhu et al’03, Bengio et al‘06]
  - Bootstrapped label propagation [Cohen‘17]
  - Competitive infection models [Rosenfeld‘17]



- Node embedding followed by vector classification
  - 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] …



Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” *IEEE Trans. Neural Nets. and Learning Systems*, 2020.

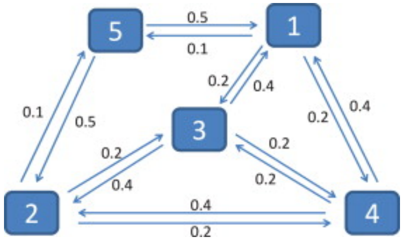
SSL through random walks on graphs

Intuition: An unlabeled node `strongly interconnected' (thus influenced) by labeled nodes of a class, likely belongs to that class

Model: Influences effected by *k*-hop paths follow *k*-step random walks (RW)

- Position X_{k-1} of walker at hop *k*-1 on node *j* wp $p_j^{(k-1)} = \Pr\{X_{k-1} = j\}$
  - Graph-guided transition probability

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



- Step-*k* landing probabilities $p_i^{(k)} := \sum_{j \in \mathcal{V}} \Pr\{X_k = i | X_{k-1} = j\} \Pr\{X_{k-1} = j\}$

$$\mathbf{p}^{(k)} = \mathbf{A}\mathbf{p}^{(k-1)} = \cdots = \mathbf{A}^k \mathbf{p}^{(0)} := [p_1^{(k)} \ldots p_N^{(k)}]^T$$

  - $[\mathbf{p}^{(k)}]_i$ measures influence of given $\mathbf{p}^{(0)}$ on node *i* after *k* steps

From landing probabilities to classification

- Max-likelihood classifier at node *i* $\hat{y}_i = \arg\max_{c \in \mathcal{Y}} \, [\mathbf{f}_c]_i$

Key idea: Model class pmfs $\mathbf{f}_c$ using landing pmfs $\{\mathbf{p}_c^{(k)}\}_{k=1}^{K}$

- Random walk per class with $\mathbf{p}_c^{(k)} = \mathbf{A}^k \mathbf{p}_c^{(0)}$ $\mathcal{L}_c := \{i \in \mathcal{L} : y_i = c\}$

  - Initial ("root") pmf
$$[\mathbf{p}_c^{(0)}]_i = \begin{cases} 1/|\mathcal{L}_c|, & i \in \mathcal{L}_c \\ 0, & \text{else} \end{cases}$$

  - Sparse ***A*** speeds up computations
$$\mathbf{P}_c^{(K)} := \left[\mathbf{p}_c^{(1)} \quad \cdots \quad \mathbf{p}_c^{(K)}\right]$$

- Weighted average of per-class landing probabilities over *K* steps
$$\mathbf{f}_c(\boldsymbol{\theta}) := \sum_{k=1}^{K} \theta_k \mathbf{p}_c^{(k)} = \mathbf{P}_c^{(K)} \boldsymbol{\theta}, \quad \boldsymbol{\theta} \in \mathcal{S}^K$$

  - Valid pmf with *K*-dim probability simplex
$$\mathcal{S}^K := \{\boldsymbol{\theta} \in \mathbb{R}^K : \; \boldsymbol{\theta} \geq \mathbf{0}, \; \mathbf{1}^\top \boldsymbol{\theta} = 1\}$$

Known members of the diffusion family

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

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

- Pmf of (class-informative) RW with restart probability 1-μ

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

- HK and PPR have fixed parameters (t, μ) that limit DoFs!

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

Adapting the diffusions

- Pmf matching with graph prior

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

$$[\mathbf{y}_{\mathcal{L}_c}]_i := \mathbb{1}\{i \in \mathcal{L}_c\}$$

  - Loss on labeled nodes

$$\ell(\mathbf{y}_{\mathcal{L}_c}, \mathbf{f}) = (|\mathcal{L}|^{-1}\mathbf{y}_{\mathcal{L}_c} - \mathbf{f})^\top \mathbf{D}_{\mathcal{L}}^{\dagger}(|\mathcal{L}|^{-1}\mathbf{y}_{\mathcal{L}_c} - \mathbf{f})$$

  - Regularizer tunes diffusion to unlabeled nodes

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

- Scalable parametrization via $\mathbf{f}_c(\boldsymbol{\theta}) = \mathbf{P}_c^{(K)}\boldsymbol{\theta}, \quad \boldsymbol{\theta}\in\mathcal{S}, \quad K \ll N$

  - **AdaDIF**

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

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

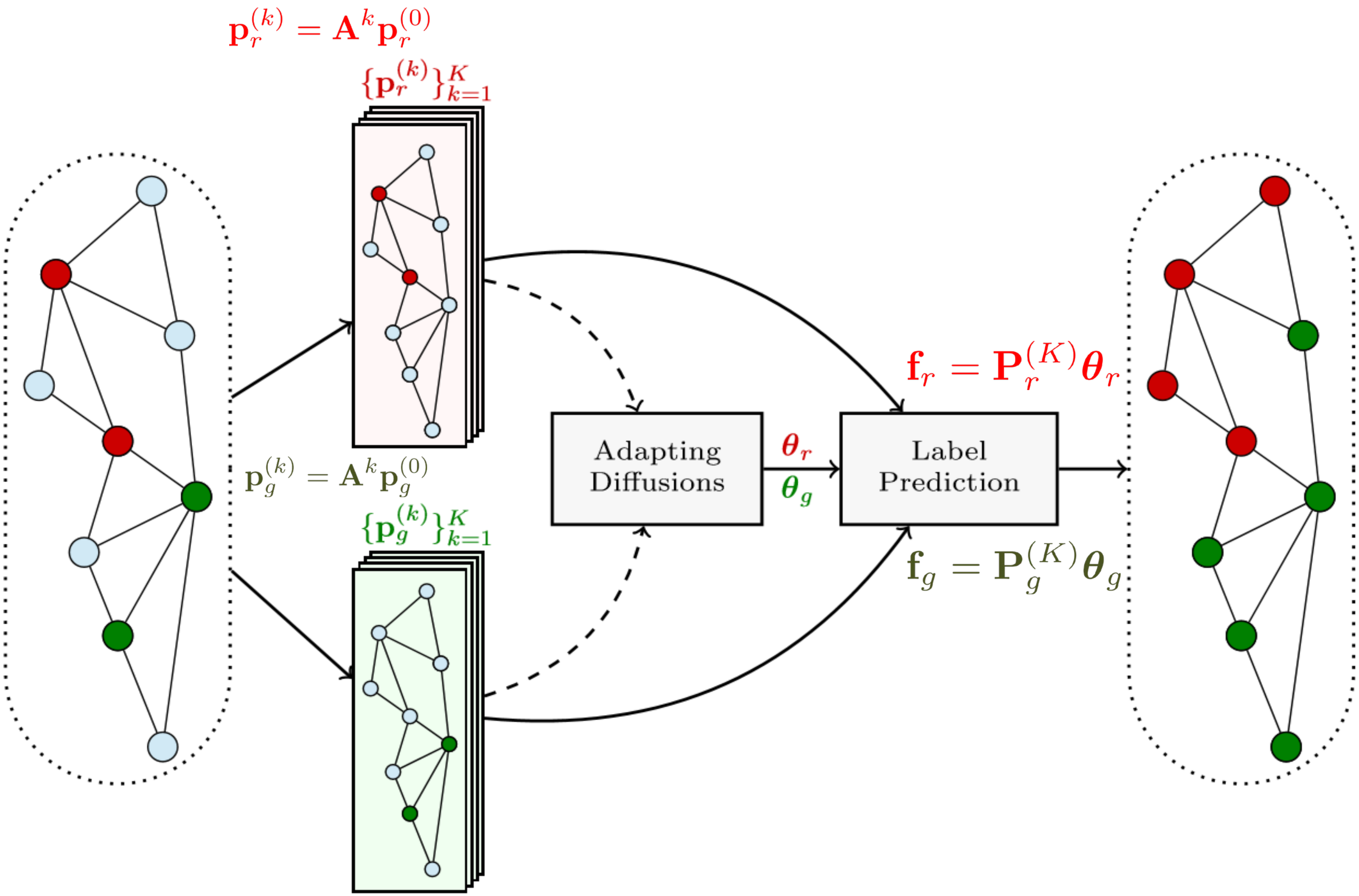
$$\mathbf{b}_c = -\frac{2}{|\mathcal{L}|}(\mathbf{P}_c^{(K)})^\top\mathbf{D}_{\mathcal{L}}^{-1}\mathbf{y}_{\mathcal{L}^c}$$

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

``Differential'' landing prob.

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

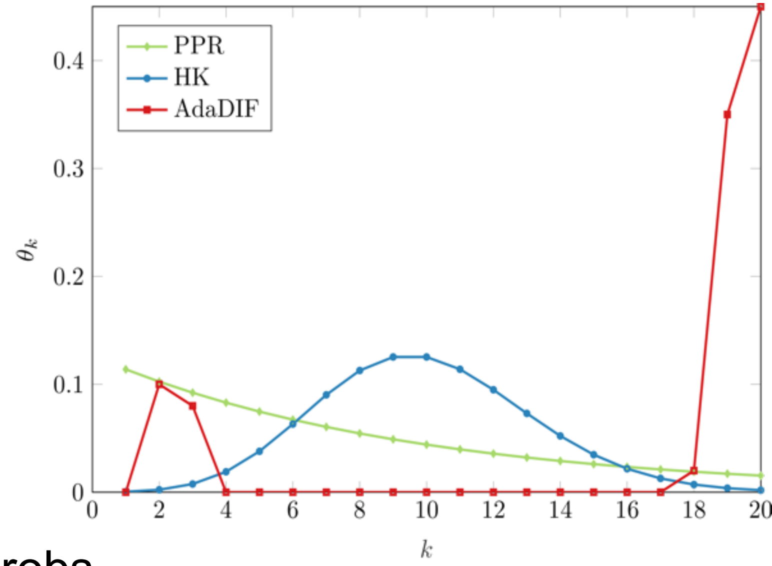
AdaDIF in a nutshell



D. K. Berberidis, A. Nikolakopoulos, and G. B. Giannakis, "Adaptive Diffusions for Scalable Learning over Graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 5, pp. 1307-1321, March 2019.

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 $\mathcal{O}(|\mathcal{E}|K)$ is low thanks to sparsity of ***A***
  - Same as non-adaptive HK and PPR; also parallelizable across classes
  - Reflect on PPR and Google … just avoid *K* >>

Constrained and unconstrained AdaDIF

- Dictionary of *D* << *K* diffusions

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

$$\mathbf{C} := \left[\mathbf{c}_1 \cdots \mathbf{c}_D\right] \in \mathbb{R}^{K \times D}$$

  - Dictionary may include PPR, HK, and more
  - Complexity $\mathcal{O}(|\mathcal{E}|(K+D))$ even when $K > |\mathcal{E}|/N$

- Unconstrained diffusions (relax simplex constraints $\theta_i \in \mathbb{R}$)
  - Retain hyperplane constraint to avoid all-zero solution
  - Closed-form solution

$$\hat{\boldsymbol{\theta}}_c = \mathbf{B}_c^{-1}(\mathbf{b}_c - \lambda^* \mathbf{1}) \qquad \lambda^* = \frac{\mathbf{1}^\top \mathbf{B}_c^{-1}\mathbf{b}_c - 1}{\mathbf{b}^\top \mathbf{B}_c^{-1}\mathbf{b}_c}$$

On the choice of K

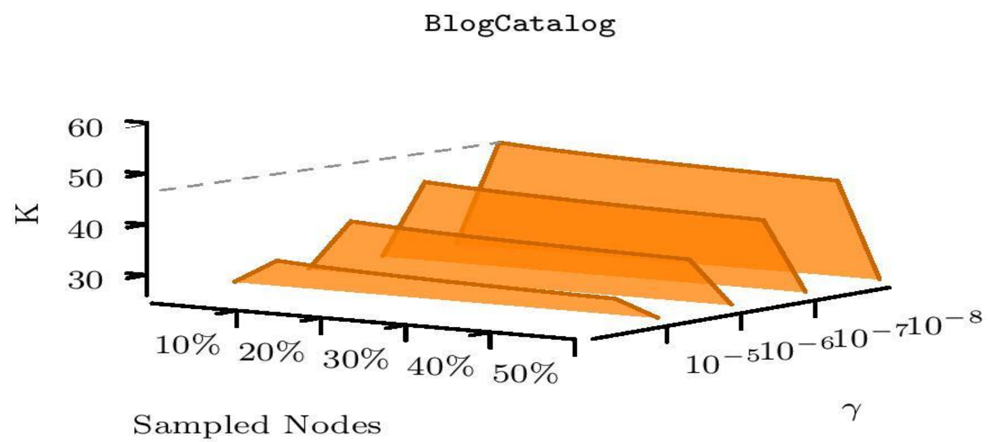
Theorem. For AdaDIF classifying two γ-distinguishable classes, K is bounded as

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

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

Take home: Too large K can compromise performance due to over-parametrization

- In a practical setup



D. K. Berberidis, A. Nikolakopoulos, and G. B. Giannakis, "Adaptive Diffusions for Scalable Learning over Graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 5, pp. 1307-1321, March 2019.

Contributions and comparisons

AdaDIF vis-à-vis graph filters [Sandryhaila-Moura‘13, Chen et al‘14]

- Different losses and regularizers, including those for outlier resilience
- Multiple class case readily addressed with AdaDIF
- AdaDIF’s simplex constraint reduces the search space
- Principled means of selecting *K* based on graph parameters

AdaDIF vis-a-vis GCNs [Atwood et al‘16], [Kipf et al‘16] … [Gama-Marques-Leus-Ribeiro’19] …

- Small number of constrained parameters: less prone to overfitting
  - Simpler and easily parallelizable training: no back propagation
- No feature inputs needed: operates naturally on graph-only settings

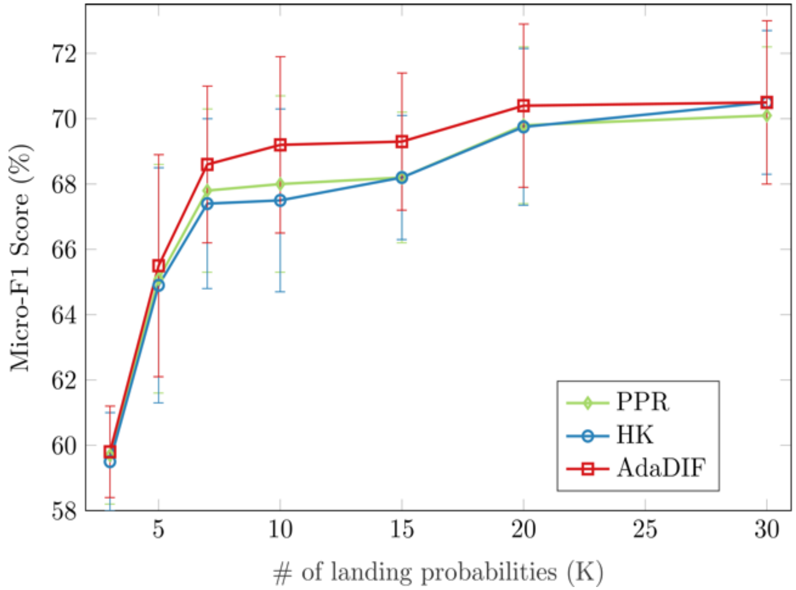
Real data tests

- Real graphs
  - Citation networks
  - Blog networks
  - Protein interaction network

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

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

- **Micro-F1**: performance influenced more by large-size classes
- **Macro-F1**: high when good performance attained across classes of variable sizes



- PPR and HK rely on *K =30* for convergence
  - AdaDIF needs just *K=15*

Multiclass graphs: Single label per node

- State-of-the-art performance with a few labeled nodes
  - Large margin improvement over the Citeseer dataset

	Graph	Cora			Citeseer			PubMed		
	$\lvert\mathcal{L}\rvert/\lvert\mathcal{V}\rvert$	2.5%	5%	10%	2.5%	5%	10%	0.25%	0.5%	1.0%
Micro-F1	AdaDIF	$\mathbf{70.5 \pm 2.4}$	$\mathbf{73.7 \pm 1.7}$	$\mathbf{77.0 \pm 1.0}$	$\mathbf{51.9 \pm 0.9}$	$\mathbf{55.1 \pm 1.0}$	$\mathbf{58.6 \pm 0.7}$	$\mathbf{72.8 \pm 2.4}$	$\mathbf{76.1 \pm 0.8}$	76.5 ± 0.5
	PPR	69.8 ± 2.5	73.3 ± 1.4	$\mathbf{77.0 \pm 1.0}$	49.7 ± 2.2	53.0 ± 1.5	57.5 ± 0.8	71.4 ± 2.6	74.4 ± 1.1	76.0 ± 0.8
	HK	70.0 ± 2.4	73.5 ± 1.8	76.7 ± 1.2	50.0 ± 2.1	53.5 ± 1.5	57.3 ± 0.9	$\mathbf{72.8 \pm 2.6}$	75.1 ± 1.0	$\mathbf{76.8 \pm 0.7}$
	Node2vec	69.5 ± 1.8	73.0 ± 1.6	75.5 ± 1.4	46.0 ± 2.7	49.7 ± 1.7	52.1 ± 1.4	$\mathbf{72.8 \pm 2.8}$	74.8 ± 1.6	75.1 ± 1.4
	Deepwalk	68.2 ± 2.5	72.1 ± 1.8	74.9 ± 1.2	45.0 ± 2.4	48.5 ± 1.7	51.2 ± 1.2	72.4 ± 2.6	73.8 ± 1.3	74.5 ± 1.2
	Planetoid-G	62.5 ± 5.1	67.3 ± 4.3	75.8 ± 1.1	43.0 ± 1.8	46.8 ± 1.9	55.2 ± 1.3	63.4 ± 3.7	65.2 ± 2.0	67.8 ± 1.5
	GCN	58.3 ± 4.0	66.5 ± 2.1	71.3 ± 1.7	38.9 ± 2.7	44.5 ± 2.0	50.3 ± 1.6	57.7 ± 3.4	64.5 ± 2.7	70.0 ± 1.5
Macro-F1	AdaDIF	$\mathbf{69.0 \pm 2.3}$	$\mathbf{72.3 \pm 1.8}$	$\mathbf{75.7 \pm 1.2}$	$\mathbf{46.6 \pm 1.1}$	$\mathbf{49.6 \pm 1.6}$	$\mathbf{53.9 \pm 1.0}$	$\mathbf{71.5 \pm 2.5}$	$\mathbf{74.2 \pm 0.7}$	75.2 ± 0.8
	PPR	66.7 ± 4.2	71.8 ± 1.6	75.3 ± 1.1	44.1 ± 2.0	48.4 ± 1.5	53.5 ± 0.8	69.5 ± 2.6	72.8 ± 1.1	74.7 ± 0.8
	HK	67.1 ± 4.2	72.1 ± 1.9	75.5 ± 1.4	44.8 ± 2.0	48.9 ± 1.5	53.7 ± 1.0	71.0 ± 2.6	73.5 ± 1.1	$\mathbf{75.6 \pm 0.8}$
	Node2vec	67.1 ± 2.6	71.6 ± 1.8	74.0 ± 1.3	42.6 ± 2.5	46.6 ± 1.7	48.7 ± 1.3	70.3 ± 3.2	73.0 ± 1.8	73.5 ± 1.4
	Deepwalk	66.1 ± 3.2	70.5 ± 2.1	73.8 ± 1.4	41.6 ± 2.4	45.5 ± 1.5	48.5 ± 1.2	70.0 ± 3.2	72.0 ± 1.7	73.1 ± 1.3
	Planetoid-G	58.0 ± 5.1	64.3 ± 4.3	74.3 ± 1.6	37.4 ± 2.1	41.6 ± 2.2	52.0 ± 2.4	61.0 ± 3.9	63.7 ± 3.0	65.2 ± 2.0
	GCN	52.0 ± 6.8	61.9 ± 2.6	64.8 ± 1.9	33.0 ± 3.0	39.2 ± 1.7	43.3 ± 1.6	52.1 ± 4.4	60.2 ± 3.9	65.3 ± 2.2

Multiclass and multilabel graphs

- Number of labels per node assumed known (typical)
  - Evaluate accuracy of top-ranking classes

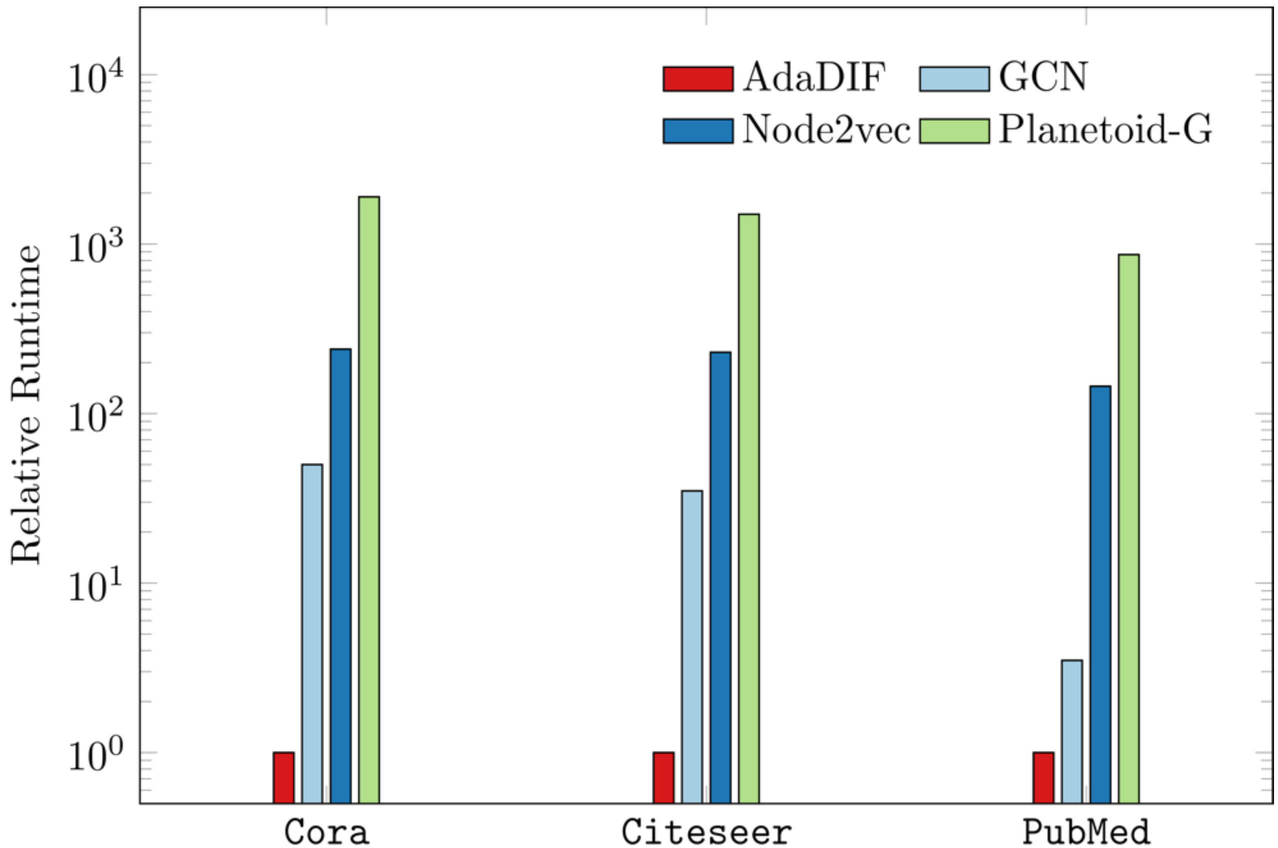
	Graph	PPI			BlogCatalog			Wikipedia		
	$\lvert\mathcal{L}\rvert/\lvert\mathcal{V}\rvert$	10%	20%	30%	10%	20%	30%	10%	20%	30%
Micro-F1	AdaDIF	15.4 ± 0.5	17.9 ± 0.7	$\mathbf{19.2 \pm 0.6}$	31.5 ± 0.6	34.4 ± 0.5	36.3 ± 0.4	28.2 ± 0.9	30.0 ± 0.5	31.2 ± 0.7
	PPR	13.8 ± 0.5	15.8 ± 0.6	17.0 ± 0.4	21.1 ± 0.8	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

- AdaDIF approaches Node2vec **Micro-F1** accuracy for PPI and BlogCatalog
  - Significant improvement over non-adaptive PPR and HK for all graphs

- Surprisingly, AdaDIF outperforms state-of-the-art **Macro-F1** performance

Runtime comparison

- AdaDIF can afford **much lower runtimes**
  - Even without parallelization!



Robust AdaDIF via leave-one-out fitting loss

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

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

- Predicted pmfs obtained via $|\mathcal{L}|$ random walks at complexity $\mathcal{O}(|\mathcal{L}|K|\mathcal{E}|)$

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

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

- Model outliers as large residuals, identify them by nnz entries of sparse $\boldsymbol{o}_c$

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

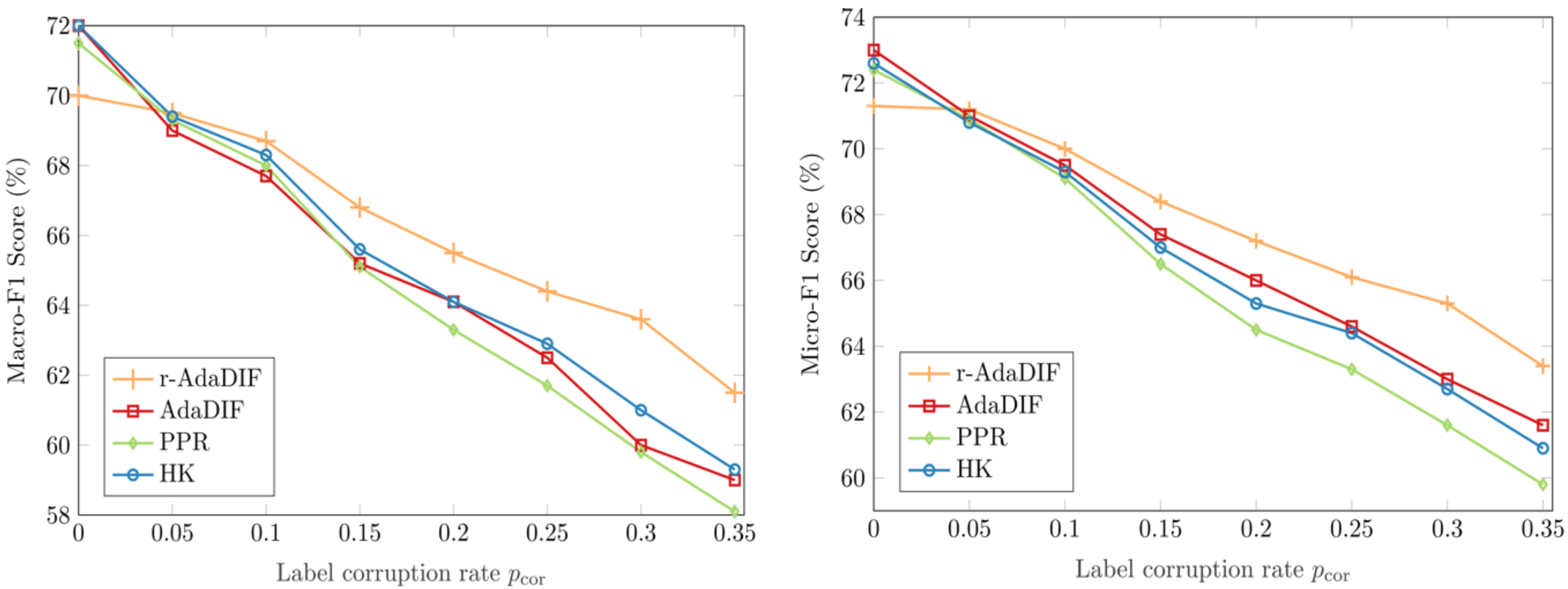
$$\hat{\boldsymbol{\theta}}_c^{(t)} = \arg\min_{\boldsymbol{\theta} \in \mathcal{S}^K} \ell^c_{\text{rob}}(\mathbf{y}_{\mathcal{L}_c} + \hat{\mathbf{o}}_c^{(t-1)}, \boldsymbol{\theta}) + \lambda_\theta \|\boldsymbol{\theta}\|_2^2 \qquad \hat{\mathbf{O}}^{(t)} = \text{SoftThres}_{\lambda_o}\left(\tilde{\mathbf{Y}}^{(t)}\right)$$

- ID and remove outliers from $\mathcal{L}$ before predicting $\mathcal{U}$

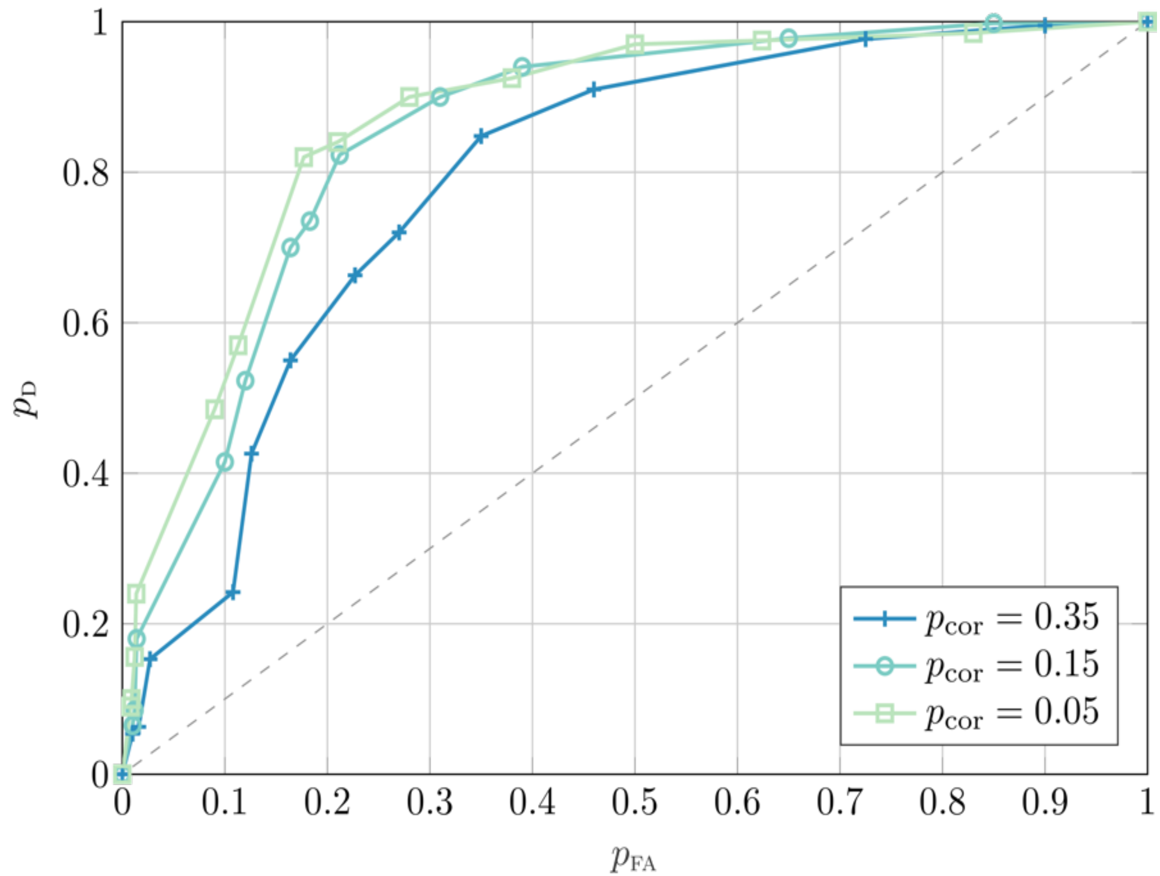
Classification performance with anomalies

- Anomalies injected in Cora graph
  - Per entry $[\mathbf{y}_{\mathcal{L}}]_i = c$, replace *c* with $c' \sim \mathrm{Unif}\{\mathcal{Y} \setminus c\}$ wp p_{cor}

- With $\lambda_o > 0$ and $p_{\mathrm{cor}} > 0$ accuracy improves after outliers are removed
  - Lower accuracy for $p_{\mathrm{cor}} = 0$ (no anomalies), since useful samples are removed

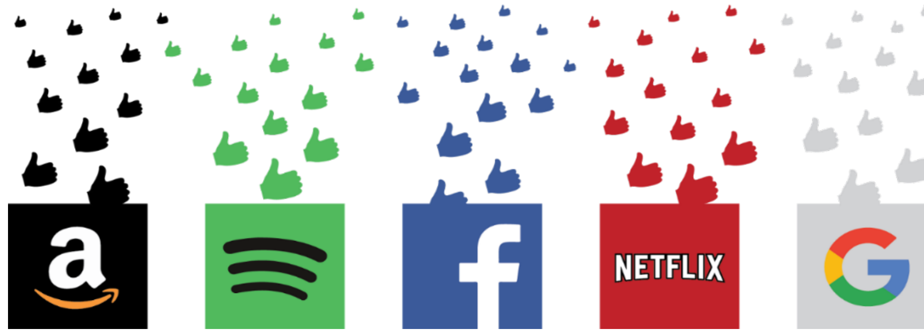


Anomaly detection performance



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

From classification to top-*R* recommendations



Goal: Given sets of *U* users, *I* items, and a *U* x *I* user-item interaction matrix ***B*** (e.g., ratings), find per user *u*, an *I* x 1 recommendation (pmf) vector $\boldsymbol{\pi}_u$

Idea: Ratings are intimately related with item-item graph connectivity

$$b_{iu} = \sum_{j \neq i} a_{ij} b_{ju} + v_{iu} \,, \quad \mathbf{b}_i = \mathbf{B}_{-i}\mathbf{a}_i + \mathbf{v}_i \,, \qquad i = 1, \ldots, I$$

- Identify item-item adjacency ***A*** (its columns in parallel!)

$$\mathbf{a}_i = \arg\min_{\mathbf{a} \in \mathfrak{R}_+} \|\mathbf{b}_i - \mathbf{B}_{-i}\mathbf{a}\|_2^2 + \gamma_1 \|\mathbf{a}\|_1 + \gamma_2 \|\mathbf{a}\|_2^2$$

- Normalize predicted ratings, and recommend *i*-th item wp $[\boldsymbol{\pi}_u]_i = \mathbf{a}_i^\top \mathbf{b}_u / \|\mathbf{b}_u\|_1$

X. Ning and G. Karypis, "SLIM: Sparse linear methods for top-*n* recommender systems," *Proc. of Intl. Conf. on Data Mining*, Vancouver, Canada, Dec. 2011, pp. 497–506.

Random walks on item-item graphs

Motivation: Go beyond neighboring interactions to broaden item-space coverage

One idea: Use $\boldsymbol{\pi}_u^{(0)} = \mathbf{b}_u / \|\mathbf{b}_u^\top\|_1$ as prior to start an ***A***-based random walk (RW)

K-step: $\boldsymbol{\pi}_u^{(K)} = \mathbf{A}^K \boldsymbol{\pi}_u^{(0)}$ PPR: $\boldsymbol{\pi}_u^{PPR} = \sum_{k=0}^{\infty} (1-\mu)\mu^k \mathbf{A}^k \boldsymbol{\pi}_u^{(0)}$

Challenge: Besides distinct initial items, users explore item-space differently

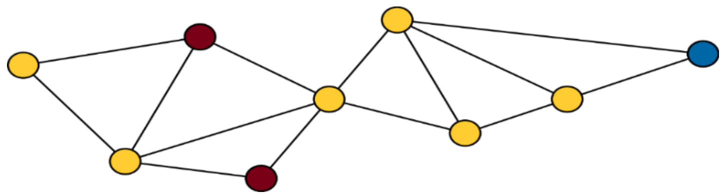
Better idea: Personalized (user-specific) item exploration process (**PerDIF**)

- Random walker *u* (with *K* biased coins) either follows ***A*** wp $\mu_u^{(1)}$ or restarts as $\boldsymbol{\pi}_u^{(0)}$ wp $1-\mu_u^{(1)}$; then tosses 2nd coin, and so on…

$$\mathbf{G}(\mu_u^{(k)}) = \mu_u^{(k)} \mathbf{A} + (1-\mu_u^{(k)}) \boldsymbol{\pi}_u^{(0)} \mathbf{1}^\top \quad \Rightarrow \quad \boxed{\boldsymbol{\pi}_u^{(K)} = \mathbf{G}(\mu_u^{(K)}) \cdots \mathbf{G}(\mu_u^{(1)}) \boldsymbol{\pi}_u^{(0)}}$$

Learning personalized restart probabilities

- Among **items rated** by user u (red and blue), choose one (blue) to assign entry=1 in target $\mathbf{h}_u \in \mathfrak{R}^{I \setminus \mathcal{N}_u + 1}$, and select also **unrated** items (yellow) with entries=0



- Learn personalized probabilities (selection matrix $\mathbf{E}_u \in \mathfrak{R}^{(I \setminus \mathcal{N}_u + 1) \times I}$

$$\boldsymbol{\mu}_u = \arg\min_{\tilde{\boldsymbol{\mu}}_u \in \mathfrak{R}^K_{(0,1)}} \| \mathbf{E}_u \mathbf{G}(\tilde{\mu}_u^{(K)}) \cdots \mathbf{G}(\tilde{\mu}_u^{(1)}) \boldsymbol{\pi}_u^{(0)} - \mathbf{h}_u \|_2^2$$

- Solve instead the convex surrogate

$$\boldsymbol{\phi}(\boldsymbol{\mu}_u) = \arg\min_{\tilde{\boldsymbol{\phi}}(\boldsymbol{\mu}_u) \in \mathcal{D}^{K+1}_{++}} \| \mathbf{E}_u \boldsymbol{\Pi}_u^{(K+1)} \tilde{\boldsymbol{\phi}}(\boldsymbol{\mu}_u) - \mathbf{h}_u \|_2^2$$

$$\boldsymbol{\phi}^\top(\boldsymbol{\mu}_u) := [1 - \mu_u^{(K)}, \mu_u^{(K)}(1 - \mu_u^{(K-1)}), \ldots, \mu_u^{(K)} \cdots \mu_u^{(2)} \mu_u^{(1)}]$$

$$\mathcal{D}^{K+1}_{++} := \{\boldsymbol{\phi} \in \mathfrak{R}^{K+1} : \boldsymbol{\phi}^\top \mathbf{1} = 1, \boldsymbol{\phi} > \mathbf{0}\} \quad \boldsymbol{\Pi}_u^{(K+1)} := [\boldsymbol{\pi}_u^{(0)}, \mathbf{A}\boldsymbol{\pi}_u^{(0)}, \ldots, \mathbf{A}^K \boldsymbol{\pi}_u^{(0)}]$$

A.N. Nikolakopoulos, D. Berberidis, G. Karypis, G. B. Giannakis, "Personalized Diffusions for Top-N Recommendation," *Proc. of ACM Intl. Conf. on Recommender Systems*, Copenhagen, Denmark, Sept. 2019.

PerDIF algorithm in a nutshell

S1. Given interactions ***B***, learn latent item-item adjacency matrix ***A***

$$\mathbf{a}_i = \arg\min_{\mathbf{a} \in \mathfrak{R}_+^C} \|\mathbf{b}_i - \mathbf{B}_{-i}\mathbf{a}\|_2^2 + \gamma_1 \|\mathbf{a}\|_1 + \gamma_2 \|\mathbf{a}\|_2^2 \,, \qquad i = 1, \ldots, I$$

S2. Rely on a modified item-item transition matrix (captures `lazy' steps)

$$\check{\mathbf{A}} = \|\mathbf{A}\|_\infty^{-1}\mathbf{A} + \mathrm{diag}(\mathbf{1} - \|\mathbf{A}\|_\infty^{-1}\mathbf{A}\mathbf{1})$$

form sparse

$$\check{\mathbf{\Pi}}_u^{(K+1)} := [\boldsymbol{\pi}_u^{(0)}, \check{\mathbf{A}}\boldsymbol{\pi}_u^{(0)}, \ldots, \check{\mathbf{A}}^K \boldsymbol{\pi}_u^{(0)}]$$

and solve for

$$\phi(\boldsymbol{\mu}_u) = \arg\min_{\tilde{\phi}(\boldsymbol{\mu}_u) \in \mathcal{D}_{++}^{K+1}} \| \, \mathbf{E}_u \check{\mathbf{\Pi}}_u^{(K+1)} \tilde{\phi}(\boldsymbol{\mu}_u) \; - \mathbf{h}_u \, \|_2^2$$

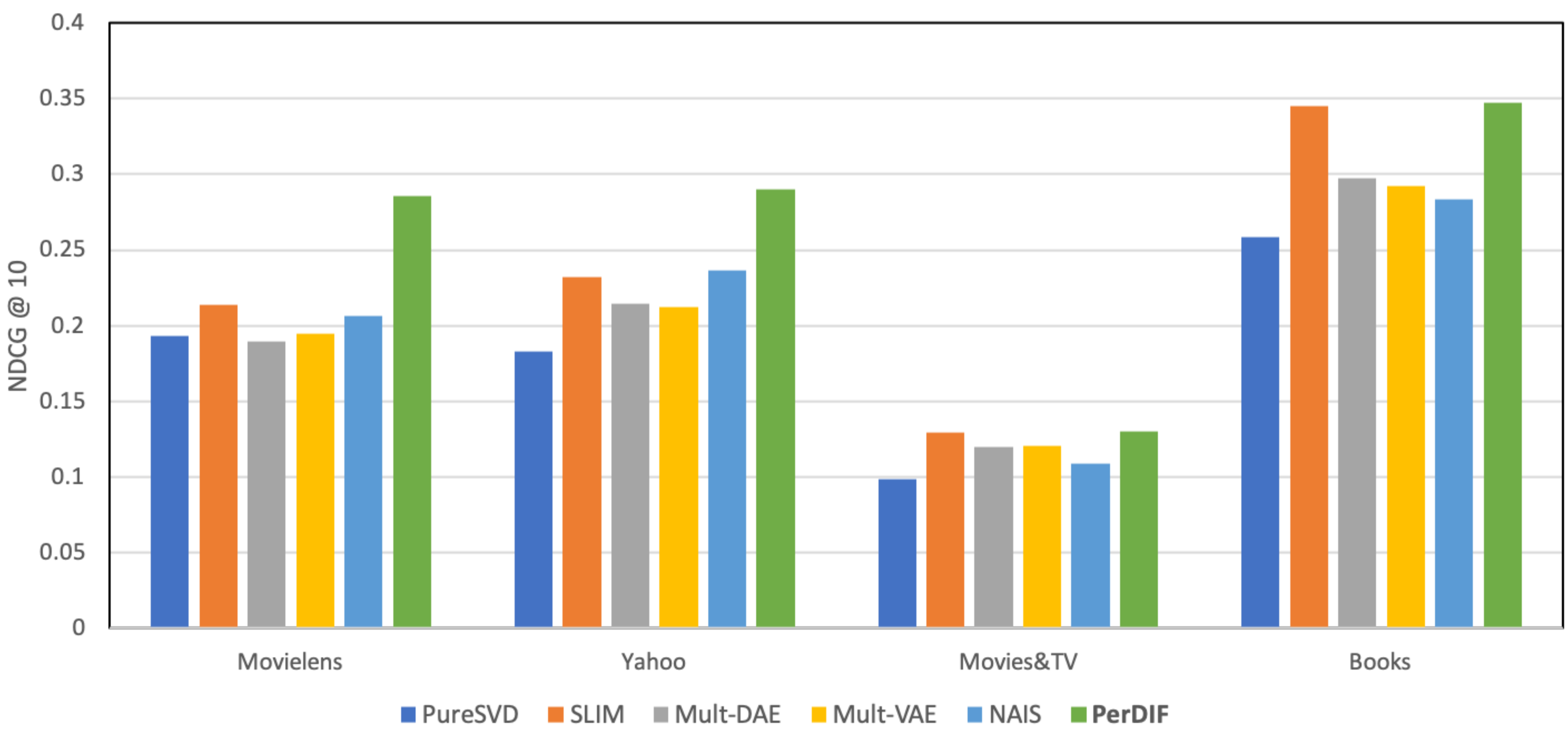
S3. Use backward substitution to obtain $\boldsymbol{\mu}_u$ with complexity $\Theta(K)$

S4. Arrive at the per-step and per-user pmf

$$\boldsymbol{\pi}_u^{(k)} = \mathbf{G}(\mu_u^{(k)}) \cdots \mathbf{G}(\mu_u^{(1)}) \boldsymbol{\pi}_u^{(0)}$$

Comparisons

- Leave-one-out protocol: 1 held-out liked item versus 999 unseen items per user



- Baselines: PureSVD [Cremonesi et al'09], SLIM [Ning-Karypis'11], Mult-DAE- Mult-VAE [Liang et al'18], NAIS [He et al'19]

PerDIF runtimes

Dataset (*UxI*)	Learning *A*	Time for all users	Time per user
Movielens (6,040x3,706)	1.4s	0.6s	0.1ms
Yahoo (7,307x3,312)	1.3s	0.4s	0.1ms
Movies&TV (10,039x5,400)	2.0s	1.3s	0.1ms
Books (43,550x24,811)	40s	55s	1.3ms
Netflix (480,189x17,770)	87m	5m	0.9ms

- Per-user fit in **milliseconds**!
  - Orders of magnitude faster than most competing baselines
  - Online tracking and adaptation to current user's needs

Current research directions

- Investigate different losses and diverse regularizers

- Further boost accuracy with nonlinear diffusion models

- Effect reduced complexity and memory requirements via approximations

- Online AdaDIF for dynamic graphs

Thank You!