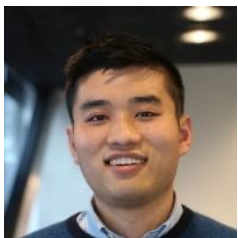
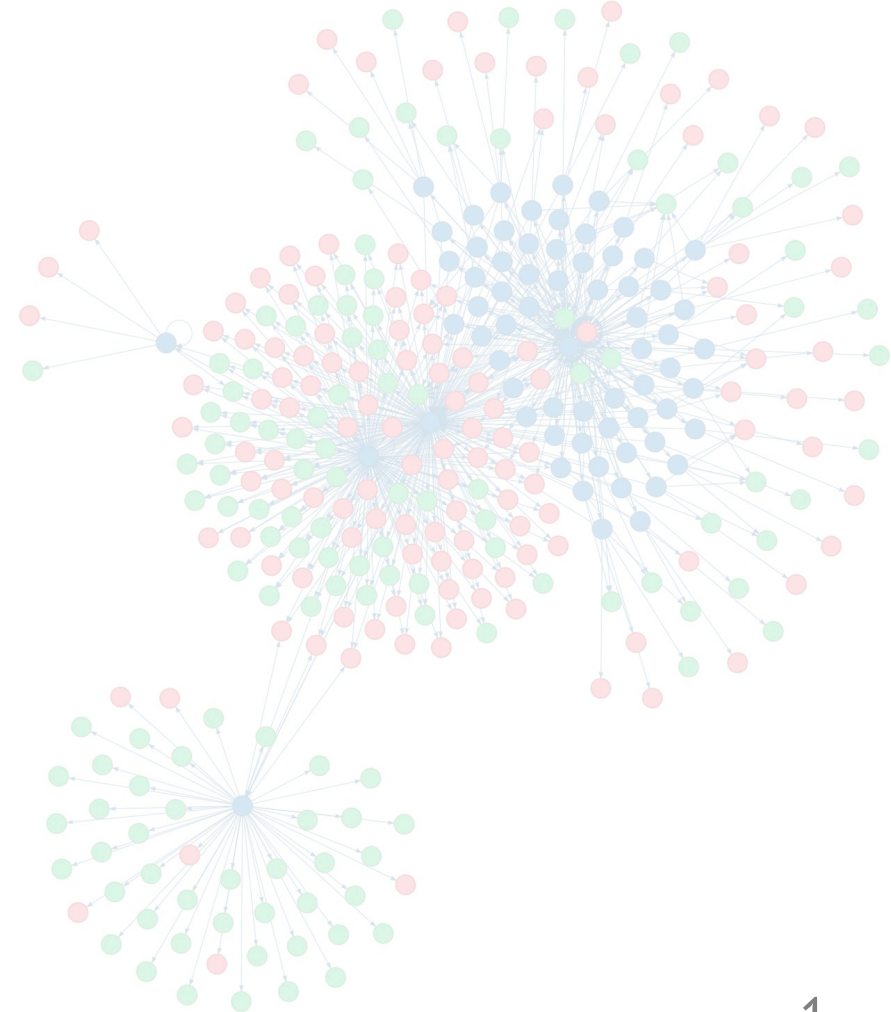


# Label Propagation and Graph Neural Networks

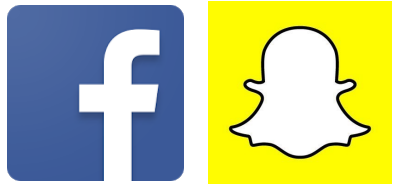
Austin Benson · Cornell University  
SIAM DM: Graph Theory and Machine Learning  
July 21, 2021



Joint work with  
Junteng Jia  
Cornell → Facebook



# Graph data modeling complex systems are everywhere.



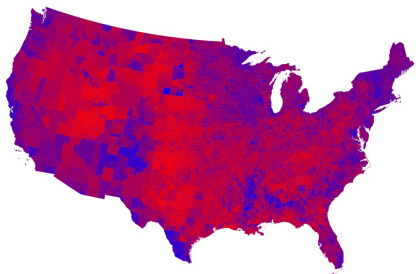
## Society

nodes are people  
edges are friendships



## Finance

nodes are accounts  
edges are transactions



## Elections

nodes are regions  
edges are social / geo

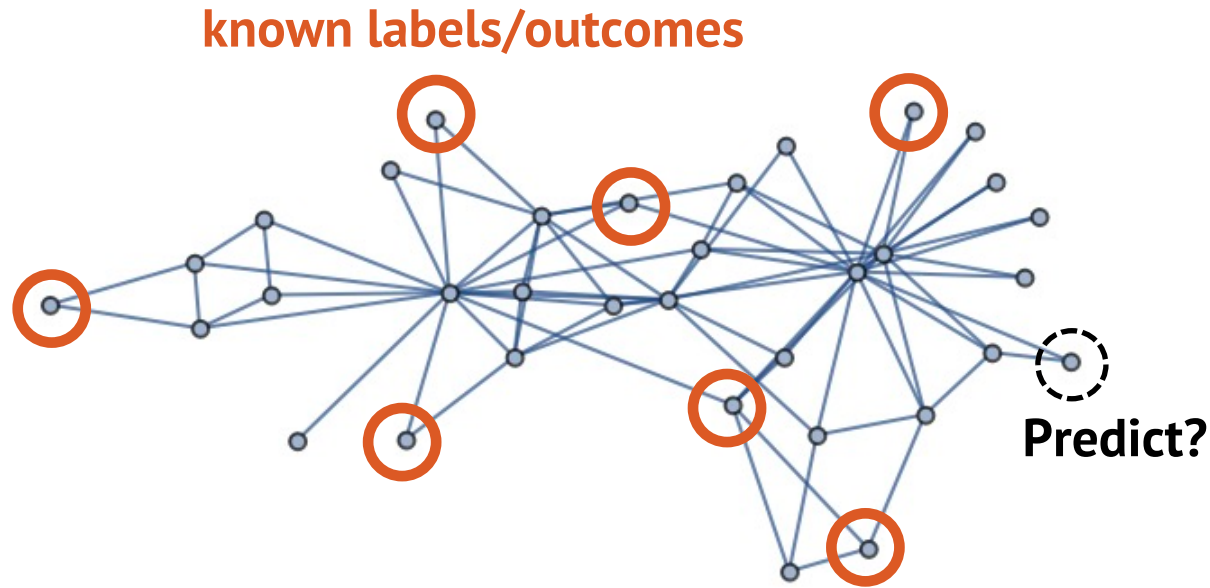
[Mark Newman 2012 map]



## Commerce

nodes are products  
edges are copurchases

# We often want to predict/estimate/construct/forecast attributes/labels/outcomes/clusters on nodes.



- Bad actors in financial transaction graphs [Weber+ 18, 19; Pareja+ 20]
- Gender in social networks [Peel 17; Altenburger–Ugander 18]
- Document classification in citation networks [Lu–Getoor 03; Kipf–Welling 17]
- Product categories from coreview/copurchase [Huang+ 20; Veldt+ 20]
- Election outcomes from social connections [Jia–Benson 21]

- Might have rich additional info on nodes (features)  
transaction history, user interests, document text, product ratings, demographics
- Graph-based semi-supervised learning, clustering, node prediction, relational learning, collective classification, community detection, ...

### Problem input.

- Graph  $G = (V, E)$ .
- $|V| \times p$  matrix  $\mathbf{X}$  of node features.
- Subset  $L \subset V$  of labeled nodes.
- Length- $|L|$  vector  $\mathbf{y}_L$  of real-valued outcomes on  $L$ .

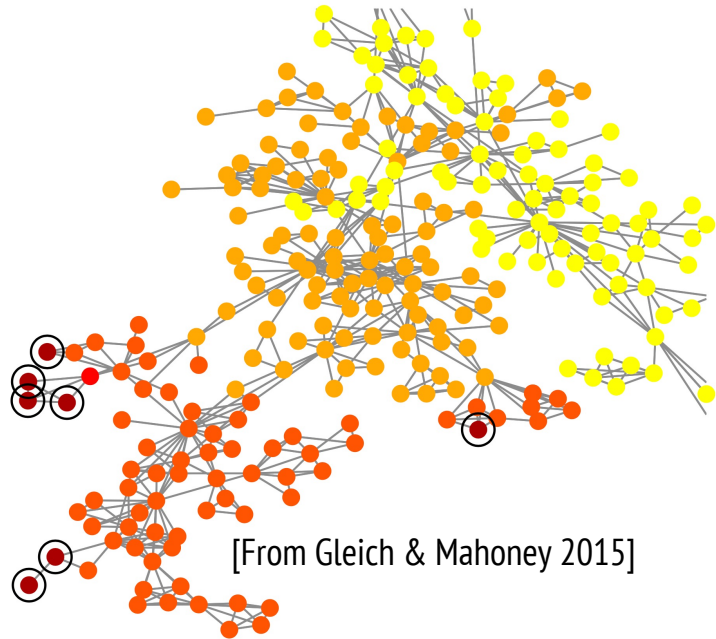
### Problem output.

- Length- $|U|$  vector  $\mathbf{y}_U$  of real-valued outcomes on  $U = V \setminus L$ .

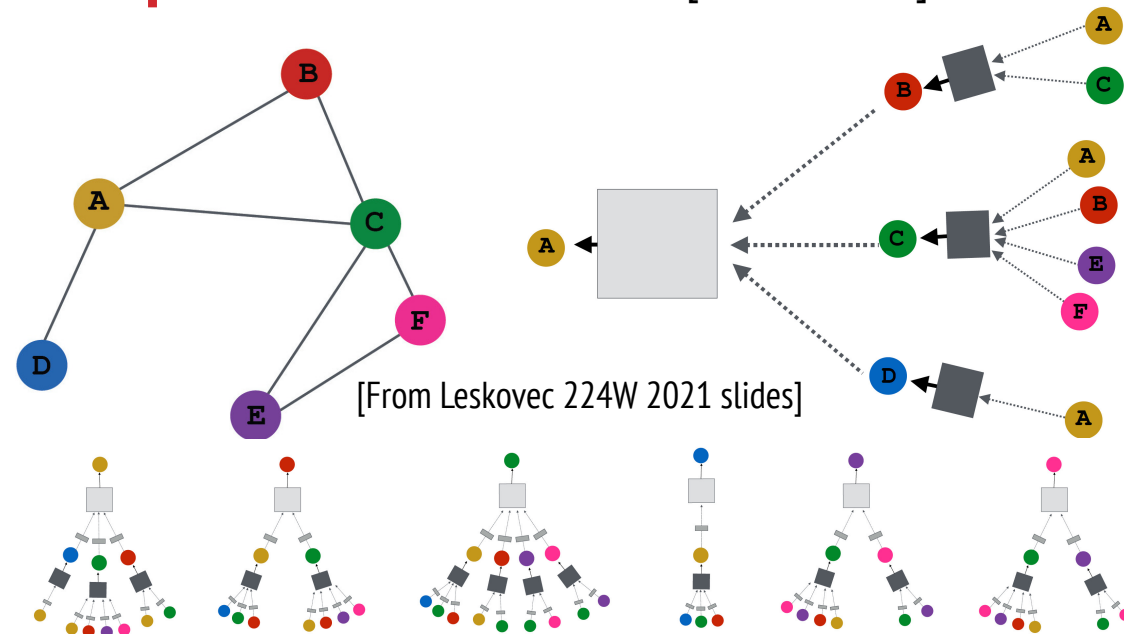


# We look at two broad classes of algorithms.

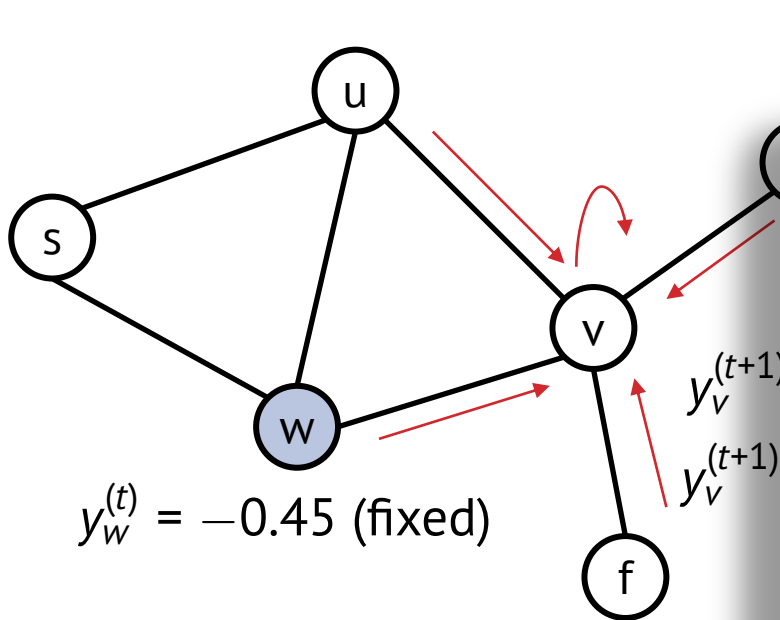
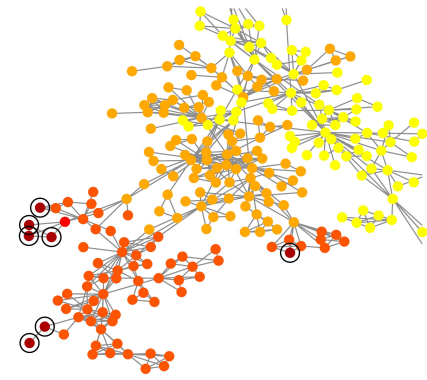
## 1. Label Propagation [early 2000s]



## 2. Graph Neural Networks [late 2010s]



# Label propagation is just neighbor averaging.



Just need SpMV

$$\mathbf{z} = \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \mathbf{y}^{(t)}$$

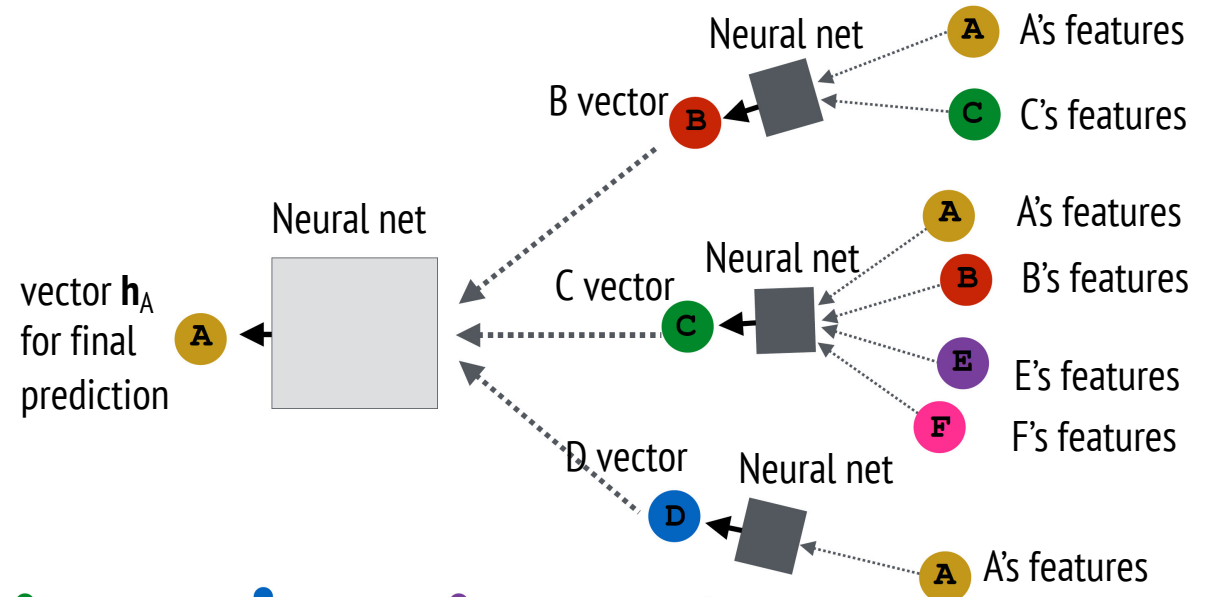
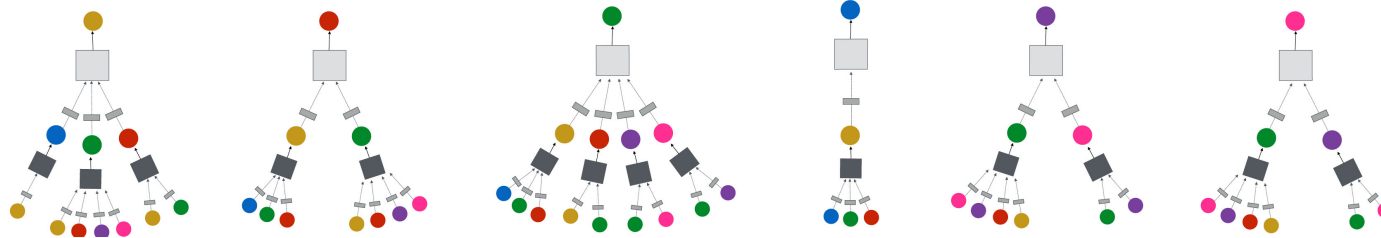
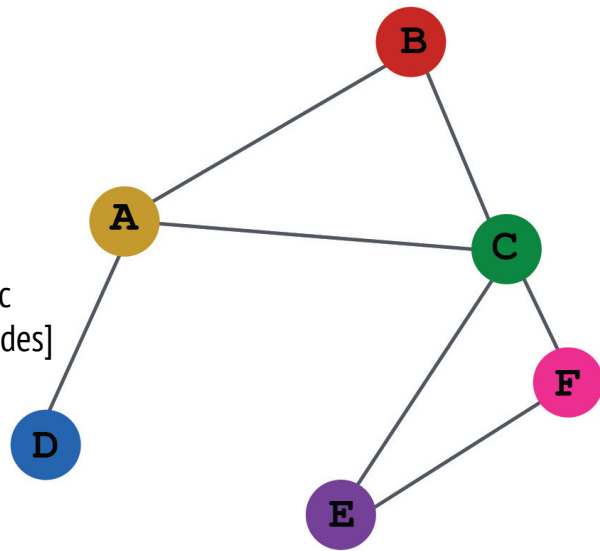
$$\mathbf{y}_U^{(t+1)} = (1 - \alpha) \mathbf{y}_U^{(0)} + \alpha \mathbf{z}_U$$

$$y_f^{(t)} + \frac{1}{\sqrt{3}} y_g^{(t)}$$

- At convergence, everyone is roughly the average over their neighbors → smooth!
- **Regression.** Start with real values (0/mean at unknown) → smoothed value for each node.

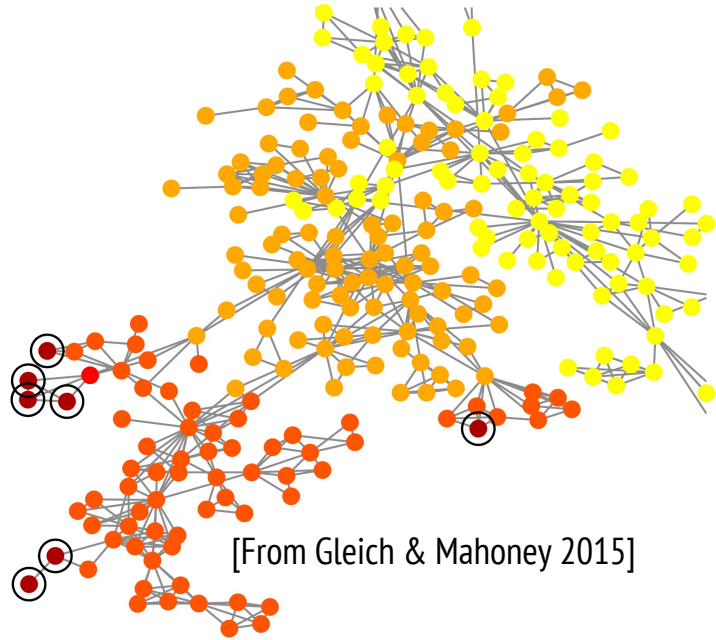
# Graph neural networks aggregate features.

[From Leskovec  
224W 2021 slides]



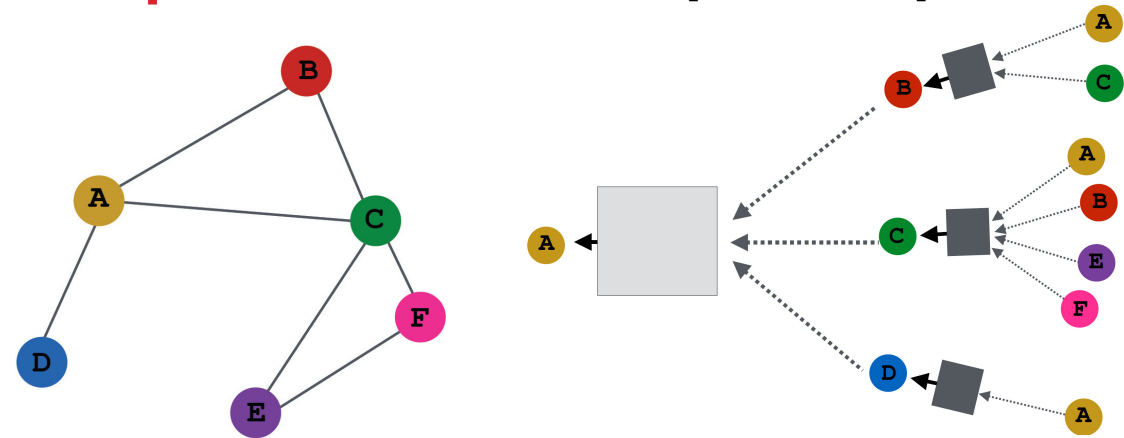
- **Regression.** Prediction at node  $A = \langle \beta, \mathbf{h}_A \rangle$ .
- **BIG** optimization problem trained with labeled nodes and automatic differentiation.
- **DIFFICULT** to implement, parallelize, reproduce.

# 1. Label Propagation [early 2000s]

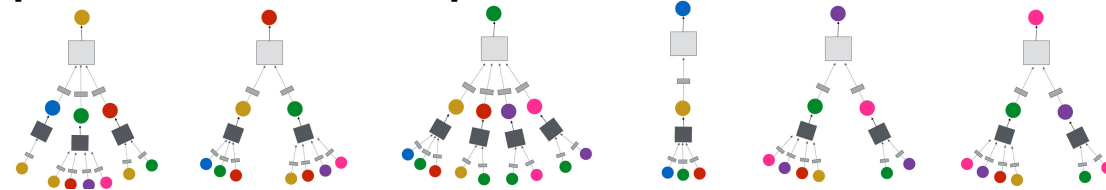


- Strong modeling assumption: connected nodes have similar labels.
- Works because of homophily [McPherson+ 01] a.k.a. assortativity [Newman 02]
- Why not use additional info/features?
- **FAST**  
a few sparse matrix-vector products

# 2. Graph Neural Networks [late 2010s]



[From Leskovec 224W 2021 slides]



- Strong modeling assumption: labels only depend on neighbor features
- Works because these features are sometimes very informative.
- Why not assume labels are correlated?
- **SLOW**  
many parameters, irregular computation

Are LP and GNNs related?

Can we avoid the complexity of GNNs?

	Node features	Neighborhood features	Neighborhood labels
Supervised ML (like OLS)	😊		
Label propagation			😊
Graph neural networks	😊	😊	
Our work	😊	😊	😊

... and it's just a few sparse matrix-vector products!

Also see *Collective Classification in Network Data* [Sen+ 08]  
for overview of similar ideas from early 2000s.

## Leaderboard for [ogbn-products](#)

The classification accuracy on the test and validation sets. The higher, the better.

Package:  $\geq 1.1.1$

Rank	Method	Test Accuracy	Validation Accuracy	Contact	References	#Params	Hardware	Date
1	<b>MLP + C&amp;S</b>	0.8418 ± 0.0007	0.9147 ± 0.0009	<a href="#">Horace He (Cornell)</a>	<a href="#">Paper</a> , <a href="#">Code</a>	96,247	GeForce RTX 2080 (11GB GPU)	Oct 27, 2020
2	<b>Linear + C&amp;S</b>	0.8301 ± 0.0001	0.9134 ± 0.0001	<a href="#">Horace He (Cornell)</a>	<a href="#">Paper</a> , <a href="#">Code</a>	10,763	GeForce RTX 2080 (11GB GPU)	Oct 27, 2020
3	<b>UniMP</b>	0.8256 ± 0.0031	0.9308 ± 0.0017	<a href="#">Yunsheng Shi (PGL team)</a>	<a href="#">Paper</a> , <a href="#">Code</a>	1,475,605	Tesla V100 (32GB)	Sep 8, 2020
4	<b>Plain Linear + C&amp;S</b>	0.8254 ± 0.0003	0.9103 ± 0.0001	<a href="#">Horace He (Cornell)</a>	<a href="#">Paper</a> , <a href="#">Code</a>	4,747	GeForce RTX 2080 (11GB GPU)	Oct 27, 2020
5	<b>DeeperGCN+FLAG</b>	0.8193 ± 0.0031	0.9221 ± 0.0037	<a href="#">Kezhi Kong</a>	<a href="#">Paper</a> , <a href="#">Code</a>	253,743	NVIDIA Tesla V100 (32GB GPU)	Oct 20, 2020

Combining Label Propagation and Simple Models Out-performs Graph Neural Networks.

Q. Huang et al., ICLR 2021.



# We developed a random model for attributes on nodes.

- Random real-valued attribute vectors  $\mathbf{a}_u = [\mathbf{x}_u; y_u]$  on each node  $u$ .
- $\mathbf{A}_i$  =  $i$ th attribute over all nodes.
- $\mathbf{N} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2}$  is the normalized Laplacian.
- Gaussian MRF random attribute model

$$\phi(\mathbf{A} | \mathbf{H}, \mathbf{h}) = \frac{1}{2} \sum_{u=1}^n \underbrace{\mathbf{a}_u^\top \mathbf{H} \mathbf{a}_u}_{\text{correlated attributes on a node}} + \frac{1}{2} \sum_{i=1}^{p+1} \underbrace{h_i \mathbf{A}_i^\top \mathbf{N} \mathbf{A}_i}_{\text{smoothness on attributes}}$$

$\mathbf{H} \in \mathbb{R}^{(p+1) \times (p+1)}$  spd,  $\mathbf{0} \leq \mathbf{h} \in \mathbb{R}^{(p+1)}$

$$= \sum_{(u,v) \in E} (A_{ui}/\sqrt{d_u} - A_{vi}/\sqrt{d_v})^2$$

$$\rho(\mathbf{A} = \mathbf{A}' | \mathbf{H}, \mathbf{h}) = \frac{e^{-\phi(\mathbf{A} | \mathbf{H}, \mathbf{h})}}{\int d\mathbf{A}' e^{-\phi(\mathbf{A}' | \mathbf{H}, \mathbf{h})}}$$

Smoother attributes are more likely (homophily / assortativity)

$$\text{vec}(\mathbf{A}) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Gamma}^{-1}), \quad \mathbf{\Gamma} = \mathbf{H} \otimes \mathbf{I}_n + \text{diag}(\mathbf{h}) \otimes \mathbf{N}$$

Just a multivariate normal random variable in the end

# Graph learning is now just statistical inference.

1. Ignore graph, condition on features  $\rightarrow$  linear regression.

$$E[\mathbf{y}|\mathbf{X} = \mathbf{X}] = \mathbf{X}^\top \boldsymbol{\beta} \rightarrow \min_{\boldsymbol{\beta}} \|\mathbf{X}_L \boldsymbol{\beta} - \mathbf{y}_L\|_2^2 \rightarrow \mathbf{X}_U \hat{\boldsymbol{\beta}}$$

**(classical derivation of linear models)**

2. Ignore features, condition on graph, labels  $\rightarrow$  label prop.

$$E[\mathbf{y}_U | \mathbf{y}_L = \mathbf{y}_L, \mathbf{G}] = -(\mathbf{I}_n + \omega \mathbf{N})_{UU}^{-1} (\mathbf{I}_n + \omega \mathbf{N})_{UL} \mathbf{y}_L, \quad \omega = h/H$$

**label prop** **Smoothing amount  $\sim$  homophily \* variance**

3. Ignore labels, condition on features + graph  $\rightarrow$  linearized GNN.

$$E[\mathbf{y} | \mathbf{X} = \mathbf{X}, \mathbf{G}] = (\mathbf{I}_n + \omega \mathbf{N})^{-1} \mathbf{X} \boldsymbol{\beta} \rightarrow \min_{\boldsymbol{\beta}} \|[(\mathbf{I}_n + \omega \mathbf{N})^{-1} \mathbf{X}]_L \boldsymbol{\beta} - \mathbf{y}_L\|_2 \rightarrow [(\mathbf{I}_n + \omega \mathbf{N})^{-1} \mathbf{X}]_U \hat{\boldsymbol{\beta}}$$

**label prop**  
**on features**

4. Condition on features + labels + graph  $\rightarrow$  linearized GNN + residual prop.

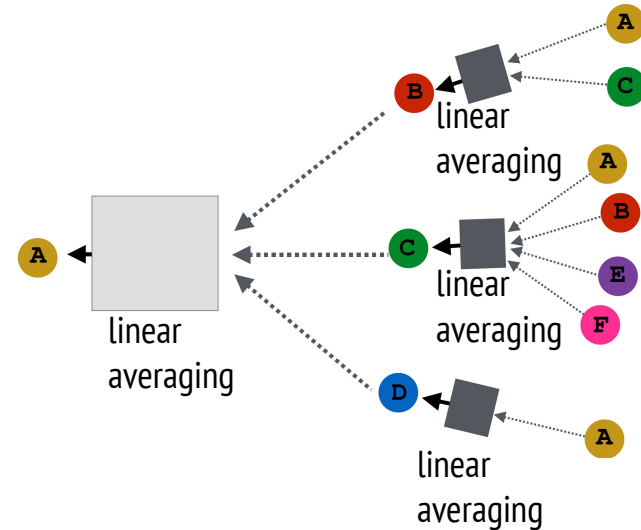
$$E[\mathbf{y}_U | \mathbf{X} = \mathbf{X}, \mathbf{y}_L = \mathbf{y}_L, \mathbf{G}] = \bar{\mathbf{y}}_U + (\mathbf{I} + \omega \mathbf{N})_{UU}^{-1} (\mathbf{I} + \omega \mathbf{N})_{UL} (\bar{\mathbf{y}}_L - \mathbf{y}_L), \quad \bar{\mathbf{y}} = (\mathbf{I}_n + \omega \mathbf{N})^{-1} \mathbf{X} \hat{\boldsymbol{\beta}}$$

**label prop** **label prop on "residuals"**  
**(on features)**

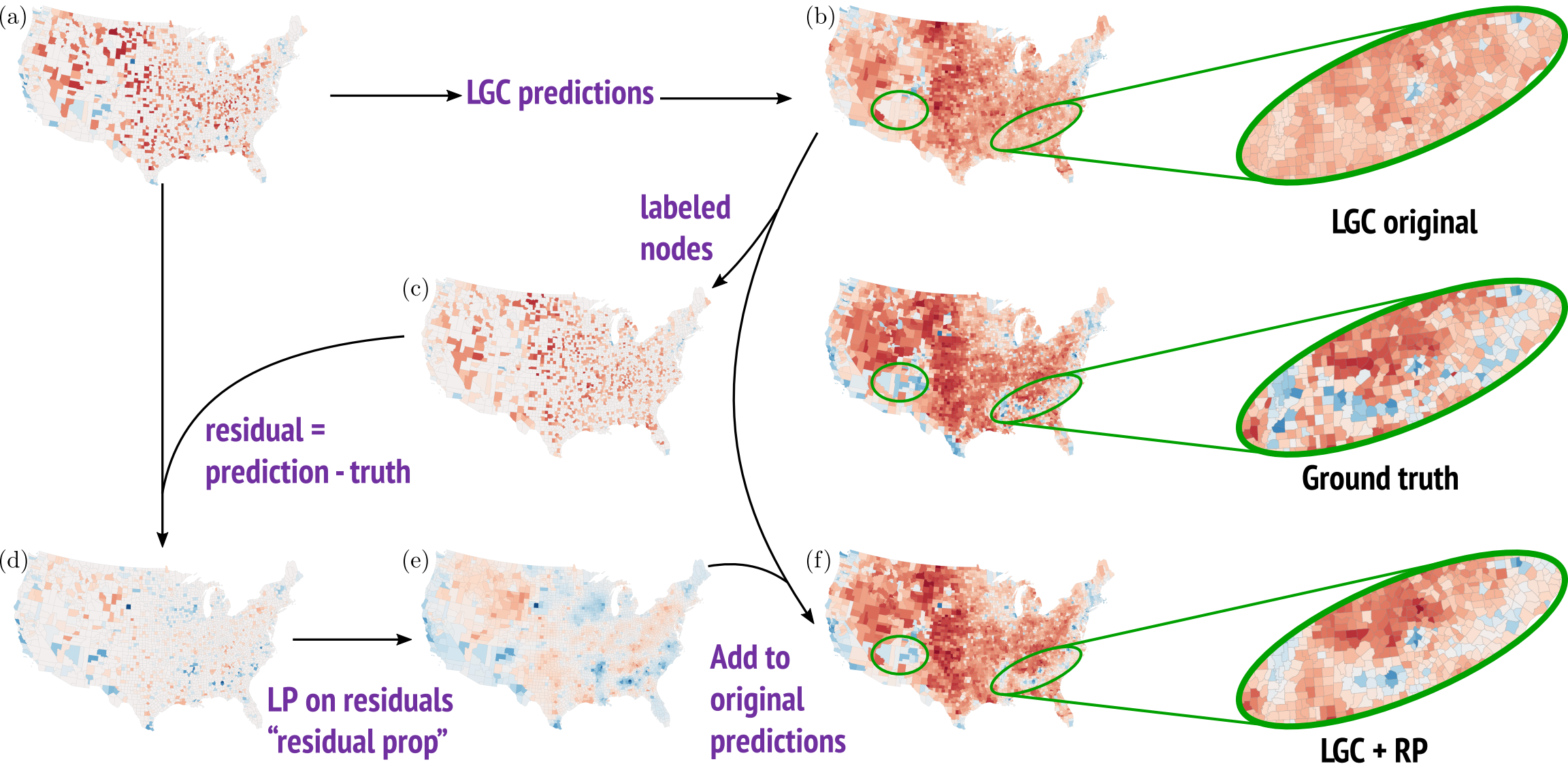
# Linear graph convolutions are linearized GNNs that come from the conditioning on features.

## Linear graph convolution (LGC).

1. Run LP on each feature  $\rightarrow$  smoothed features.
2. Ordinary least squares on these preprocessed, smoothed features.



# Residual propagation smooths errors.



Dataset	Outcome	LP	LR	LGC	SGC	GCN	LGC/RP
USA counties	election	0.52	0.42	0.49	0.43	0.52	<b>0.64</b>
	unemployment	0.47	0.34	0.39	0.32	0.45	<b>0.54</b>
climate	landT	0.89	0.81	0.81	0.79	<b>0.91</b>	0.90
	pm2.5	0.96	0.21	0.27	0.23	0.78	<b>0.96</b>
Twitch	days	0.08	0.58	0.59	0.22	0.26	<b>0.60</b>

```

1  function LGC_params(S, X, y, L;  $\alpha=0.9$ , num_iters=10)
2      X_smooth = copy(X)
3      for _ in 1:num_iters
4          X_smooth = (1 -  $\alpha$ ) * X +  $\alpha$  * S * X_smooth
5      end
6      return X_smooth, X_smooth[L, :] \ y[L]
7  end
8
9  function residual_prop(S, y,  $\bar{y}$ , U;  $\alpha=0.9$ , num_iters=10)
10     r = y -  $\bar{y}$ 
11     r[U] = 0
12     for _ in 1:num_iters
13         z = S * r
14         r[U] =  $\alpha$  * z[U]
15     end
16     return r
17 end
18
19 function LGC_RP_prediction(
20     S, # normalized adjacency  $D^{-1/2} A D^{-1/2}$ 
21     X, # n x d feature matrix for n nodes
22     U, # indices of unlabeled nodes
23     L # indices of labeled nodes
24     y, # n x 1 label vector (zero on y[U])
25 )
26     X_smooth,  $\hat{\beta}$  = LGC_params(S, X, y, L)
27      $\bar{y}$  = X_smooth *  $\hat{\beta}$ 
28     r = residual_prop(S, y,  $\bar{y}$ , L)
29     return  $\bar{y}$ [U] + r[U]
30 end

```

# Linear graph convolutions are linearized GNNs that come from the conditioning on features.

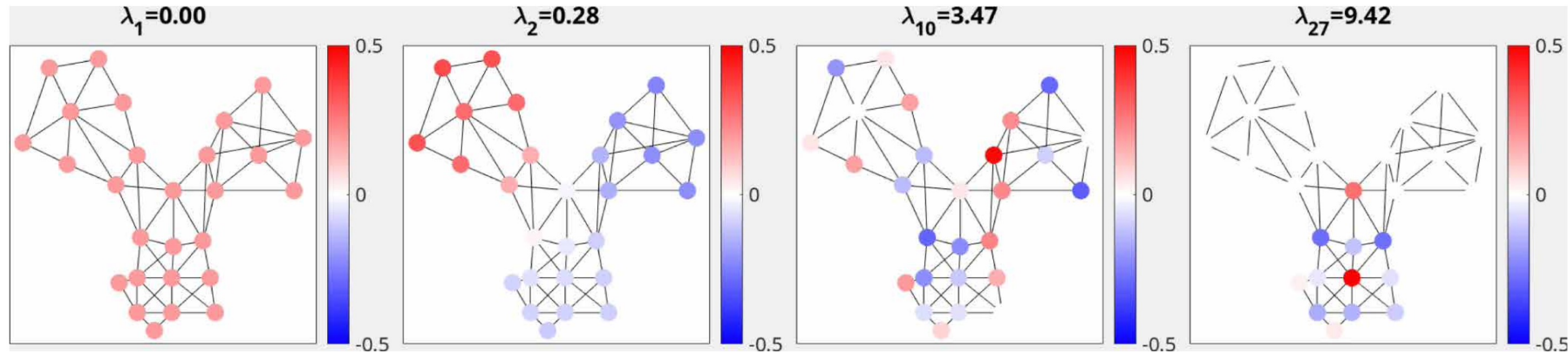
**Linear Graph Convolution (LGC)**  $(1 - \alpha) (I + \alpha S + \alpha^2 S^2 + \dots) X \beta$   $S = D^{-1/2} W D^{-1/2}$   
[Jia-Benson 21]

**Simplified Graph Convolution (SGC)**  $\tilde{S}^K X \beta$   $\tilde{S} = (D + I)^{-1/2} (W + I) (D + I)^{-1/2}$   
[Wu+ 19]

**Graph Convolution Network (GCN)**  $\sigma(\tilde{S} \dots \sigma(\tilde{S} X \Theta^{(1)}) \dots \Theta^{(K)}) \beta$   
[Kipf-Welling 17]



# Our model helps us understand smoothing.



Graph Signal Processing: Overview, Challenges and Applications, Ortega et al., Proc. IEEE, 2018.

$$\mathbf{N} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \text{ feature } \mathbf{f} = \sum_{i=1}^n c_i \mathbf{V}_i$$

$$\text{LGC } \mathbf{f} \rightarrow \sum_{i=1}^n \frac{1}{(1 + \omega \lambda_i)} c_i \mathbf{V}_i$$

Low-pass on  $[0, \infty)$ ,  
continuous parameterization.

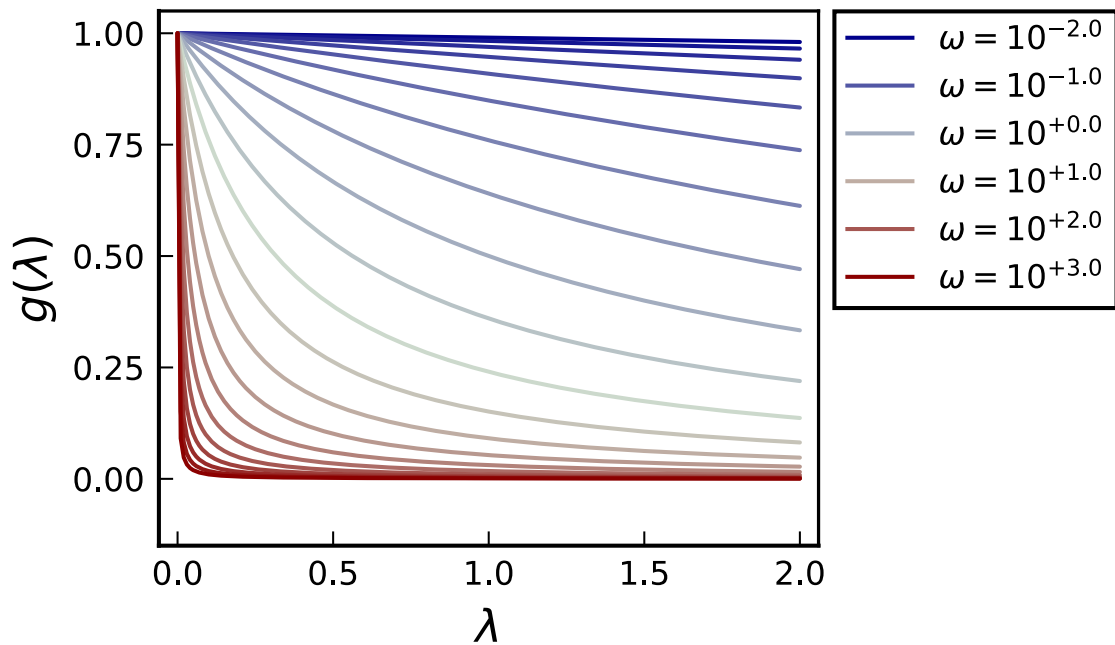
$$\text{SGC } \mathbf{f} \rightarrow \sum_{i=1}^n (1 - d/(d+1)\lambda_i)^K c_i \mathbf{V}_i$$

Low-pass on  $[0, (d + 1)/d]$ ,  
discrete parameterization.

Encouraging  
smoothness.

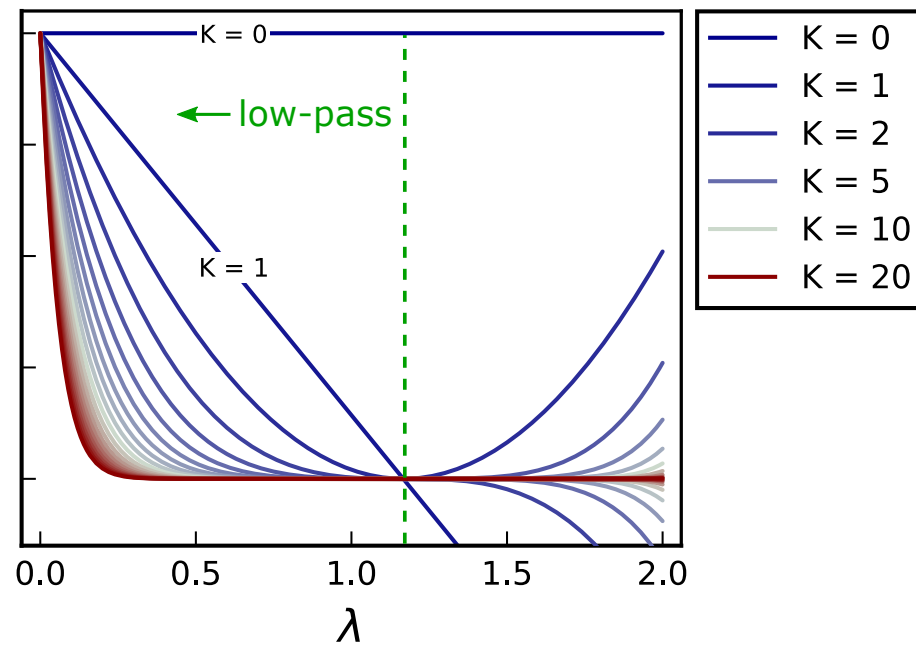
# Our model helps us understand smoothing.

LGC



$$\mathbf{f} = \sum_{i=1}^n c_i \mathbf{V}_i \rightarrow \sum_{i=1}^n \frac{1}{(1 + \omega \lambda_i)} c_i \mathbf{V}_i$$

SGC



$$\mathbf{f} = \sum_{i=1}^n c_i \mathbf{V}_i \rightarrow \sum_{i=1}^n (1 - d/(d+1)\lambda_i)^K c_i \mathbf{V}_i$$

# Our model provides a nice setup for inductive learning.

## Problem input.

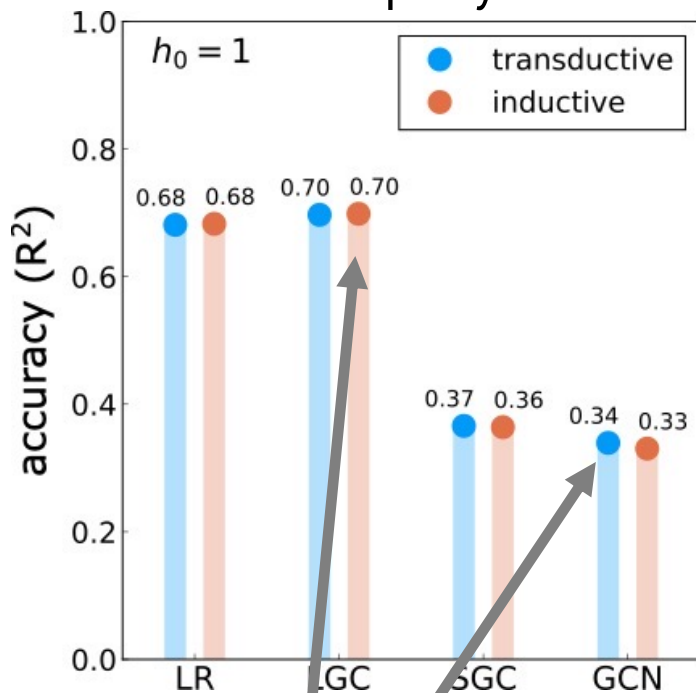
- Graphs  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$ .
- $|V_1| \times p$  matrix  $\mathbf{X}_1$  and  $|V_2| \times p$  matrix  $\mathbf{X}_2$  of node features (same features)
- Subset  $L_1 \subset V$  of labeled nodes.
- Length- $|L_1|$  vector  $\mathbf{y}_{L_1}$  of outcomes on  $L_1$ .

## Problem output.

- Length- $|V_2|$  vector  $\mathbf{y}$  of outcomes on nodes  $V_2$ .

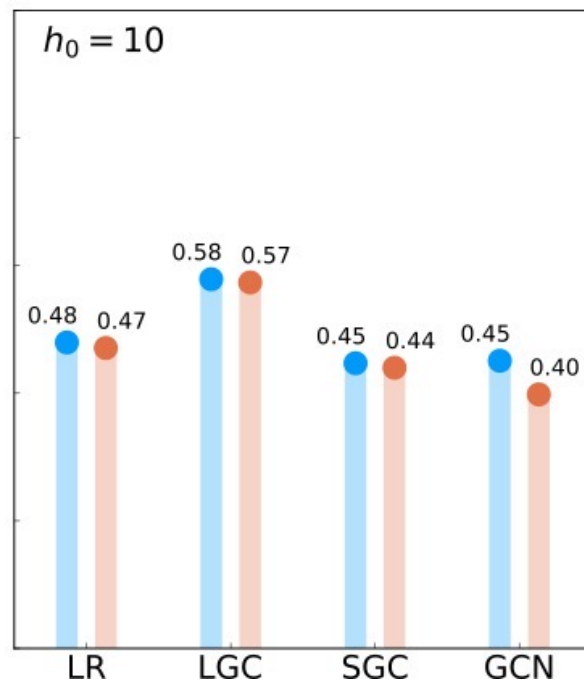
# Our model provides a nice setup for inductive learning.

Predictive features,  
low homophily.

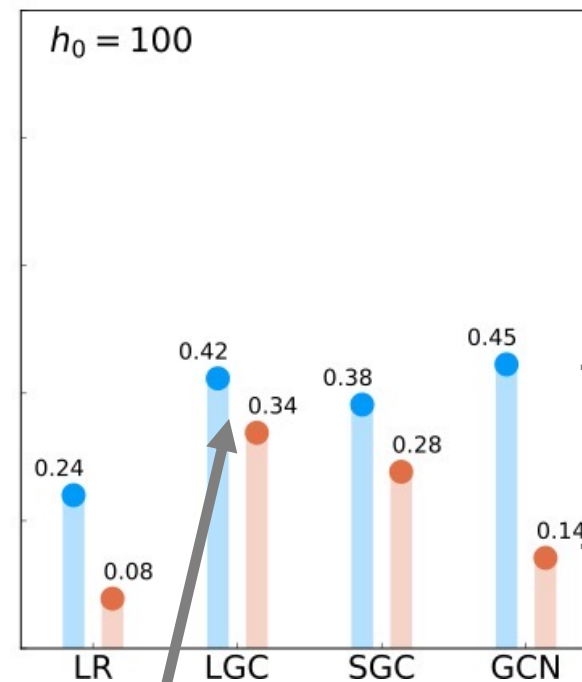


No performance  
degradation.

High homophily.

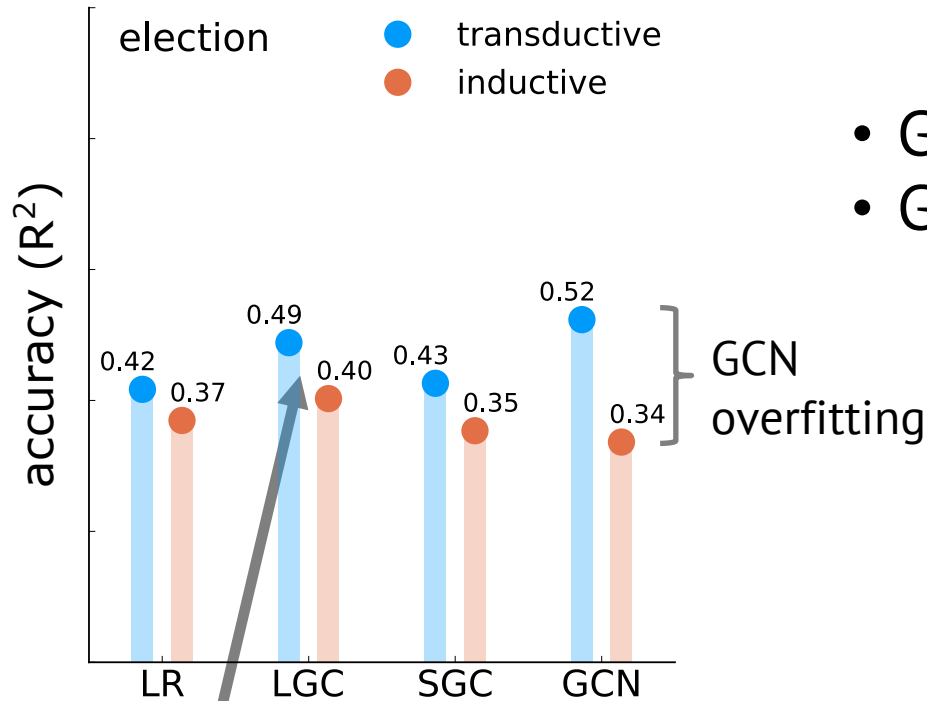


A bit of degradation.



Bad  
overfitting!

# Our model provides a nice setup for inductive learning.



- Graph  $G_1$  from 2012 election data.
- Graph  $G_2$  from 2016 election data.



A bit of degradation.

# Label propagation is a powerful tool.



1. LP can be applied to features (smoothing / de-noising).
2. LP can be applied to residuals (correlated errors).
3. While traditionally seen as separate ideas, LP and basic GNN ideas can be derived from a common model and combined effectively.
4. LP is scalable and easy to program.
5. Linear models are often superior to nonlinear ones (GNNs) in practice... you just need to find the right one.

$$\mathbf{y}_U^{\text{LGC/RP}} = [(\mathbf{I}_n + \omega \mathbf{N})^{-1} \mathbf{X} \boldsymbol{\beta}]_U - (\mathbf{I} + \omega \mathbf{N})_{UU}^{-1} (\mathbf{I} + \omega \mathbf{N})_{UL} (\mathbf{y}_L - [(\mathbf{I}_n + \omega \mathbf{N})^{-1} \mathbf{X} \boldsymbol{\beta}]_L)$$

# Label Propagation and Graph Neural Networks



**THANKS!** Austin R. Benson  
<http://cs.cornell.edu/~arb>  
 @austinbenson  
 arb@cs.cornell.edu

A Unifying Generative Model for Graph Learning Algorithms: Label Propagation, Graph Convolutions, and Combinations.  
Junteng Jia and Austin R. Benson. arXiv:2101.07730, 2021.

  [github.com/000Justin000/GaussianMRF](https://github.com/000Justin000/GaussianMRF)



Graph Belief Propagation Networks.

Junteng Jia, Cenk Baykal, Vamsi K. Potluru, and Austin R. Benson. arXiv:2106.03033, 2021.

  [github.com/000Justin000/GBPN](https://github.com/000Justin000/GBPN)



Residual Correlation in Graph Neural Network Regression.

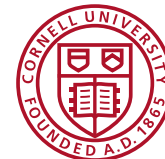
Junteng Jia and Austin R. Benson. Proc. of KDD, 2020.

  [github.com/000Justin000/gnn-residual-correlation](https://github.com/000Justin000/gnn-residual-correlation)

Combining Label Propagation and Simple Models Out-performs Graph Neural Networks.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R. Benson. Proc. of ICLR, 2021.

  [github.com/CUAI/CorrectAndSmooth](https://github.com/CUAI/CorrectAndSmooth)



Cornell University