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

**Elections**
- nodes are regions
- edges are social / geo

**Finance**
- nodes are accounts
- edges are transactions

**Commerce**
- nodes are products
- edges are copurchases

[Mark Newman 2012 map]
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.

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.

Just need SpMV

\[
\begin{align*}
z &= D^{-1/2} AD^{-1/2} y^{(t)} \\
y_U^{(t+1)} &= (1 - \alpha)y_U^{(0)} + \alpha z_U \\
y_f^{(t)} &= y_g^{(t)}
\end{align*}
\]
Graph neural networks aggregate features.

• **Regression.** Prediction at node $A = \langle \beta, h_A \rangle$.
• **BIG** optimization problem trained with labeled nodes and automatic differentiation.
• **DIFFICULT** to implement, parallelize, reproduce.

[From Leskovec 224W 2021 slides]
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]

- 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?
<table>
<thead>
<tr>
<th></th>
<th>Node features</th>
<th>Neighborhood features</th>
<th>Neighborhood labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised ML (like OLS)</td>
<td>😊</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label propagation</td>
<td>😊</td>
<td></td>
<td>😊</td>
</tr>
<tr>
<td>Graph neural networks</td>
<td>😊</td>
<td>😊</td>
<td></td>
</tr>
<tr>
<td>Our work</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
</tr>
</tbody>
</table>

... 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: >=1.1.1**

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

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 \( \mathbf{a}_u = [x_u, y_u] \) on each node \( u \).
- \( \mathbf{A}_i = \) ith attribute over all nodes.
- \( \mathbf{N} = I - \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2} \) is the normalized Laplacian.
- Gaussian MRF random attribute model

\[
\begin{align*}
\phi(\mathbf{A} \mid \mathbf{H}, \mathbf{h}) &= \frac{1}{2} \sum_{u=1}^{n} \mathbf{a}_u^T \mathbf{H} \mathbf{a}_u + \frac{1}{2} \sum_{i=1}^{p+1} \mathbf{h}_i \mathbf{A}_i^T \mathbf{N} \mathbf{A}_i, \quad \mathbf{H} \in \mathbb{R}^{(p+1) \times (p+1)} \text{ spd}, \quad 0 \leq \mathbf{h} \in \mathbb{R}^{(p+1)} \\
\rho(\mathbf{A} = \mathbf{A} \mid \mathbf{H}, \mathbf{h}) &= \frac{e^{-\phi(\mathbf{A} \mid \mathbf{H}, \mathbf{h})}}{\int d\mathbf{A}' \; e^{-\phi(\mathbf{A}' \mid \mathbf{H}, \mathbf{h})}} \quad \text{Smotherer attributes are more likely (homophily / assortativity)}
\end{align*}
\]

\[\text{vec}(\mathbf{A}) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Gamma}^{-1}), \quad \mathbf{\Gamma} = \mathbf{H} \otimes \mathbf{I}_n + \text{diag}(\mathbf{h}) \otimes \mathbf{N} \quad \text{Just a multivariate normal random variable in the end}\]
Graph learning is now just statistical inference.

1. Ignore graph, condition on features $\rightarrow$ linear regression.
   
   \[ E[y|X = x] = X^T \beta \quad \rightarrow \quad \min_{\beta} \| X_L \beta - y_L \|_2^2 \quad \rightarrow \quad X_U \hat{\beta} \]
   
   (classical derivation of linear models)

2. Ignore features, condition on graph, labels $\rightarrow$ label prop.
   
   \[ E[y_U|y_L = y_L, G] = -(I_n + \omega N)^{-1} (I_n + \omega N)_{UL} y_L, \quad \omega = h/H \]
   
   label prop
   
   Smoothing amount $\sim$ homophily $\ast$ variance

3. Ignore labels, condition on features + graph $\rightarrow$ linearized GNN.
   
   \[ E[y|X = x, G] = (I_n + \omega N)^{-1} X \beta \quad \rightarrow \quad \min_{\beta} \| [(I_n + \omega N)^{-1} X]_L \beta - y_L \|_2 \quad \rightarrow \quad [(I_n + \omega N)^{-1} X]_U \hat{\beta} \]
   
   label prop on features

4. Condition on features + labels + graph $\rightarrow$ linearized GNN + residual prop.
   
   \[ E[y_U|X = x, y_L = y_L, G] = \bar{y}_U + (I + \omega N)^{-1} (I + \omega N)_{UL} (\bar{y}_L - y_L), \quad \bar{y} = (I_n + \omega N)^{-1} X \hat{\beta} \]
   
   label prop (on features)
   
   label prop on “residuals”
Linear graph convolutions are linearized GNNs that come from the conditioning on features.

**Linear graph convolution (LGC).**
1. Run LP on each feature → smoothed features.
2. Ordinary least squares on these preprocessed, smoothed features.
Residual propagation smooths errors.

LGC predictions

residual = prediction - truth

LGC original

Ground truth

LGC + RP

LP on residuals “residual prop”

Add to original predictions

labeled nodes
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Outcome</th>
<th>LP</th>
<th>LR</th>
<th>LGC</th>
<th>SGC</th>
<th>GCN</th>
<th>LGC/RP</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA counties</td>
<td>election</td>
<td>0.52</td>
<td>0.42</td>
<td>0.49</td>
<td>0.43</td>
<td>0.52</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td>unemployment</td>
<td>0.47</td>
<td>0.34</td>
<td>0.39</td>
<td>0.32</td>
<td>0.45</td>
<td>0.54</td>
</tr>
<tr>
<td>climate</td>
<td>landT</td>
<td>0.89</td>
<td>0.81</td>
<td>0.81</td>
<td>0.79</td>
<td>0.91</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>pm2.5</td>
<td>0.96</td>
<td>0.21</td>
<td>0.27</td>
<td>0.23</td>
<td>0.78</td>
<td>0.96</td>
</tr>
<tr>
<td>Twitch</td>
<td>days</td>
<td>0.08</td>
<td>0.58</td>
<td>0.59</td>
<td>0.22</td>
<td>0.26</td>
<td>0.60</td>
</tr>
</tbody>
</table>
function LGC_params(S, X, y, L; α=0.9, num_iters=10)
    X_smooth = copy(X)
    for _ in 1:num_iters
        X_smooth = (1 - α) * X + α * S * X_smooth
    end
    return X_smooth, X_smooth[L, :] \ y[L]
end

function residual_prop(S, y, ŷ, U; α=0.9, num_iters=10)
    r = y - ŷ
    r[U] = 0
    for _ in 1:num_iters
        z = S * r
        r[U] = α * z[U]
    end
    return r
end

function LGC_RP_prediction(
    S,  # normalized adjacency D^{-1/2} A D^{-1/2}
    X,  # n x d feature matrix for n nodes
    U,  # indices of unlabeled nodes
    L,  # indices of labeled nodes
    y,  # n x 1 label vector (zero on y[U])
)
    X_smooth, Œ = LGC_params(S, X, y, L)
    ŷ = X_smooth * Œ
    r = residual_prop(S, y, ŷ, L)
    return ŷ[U] + c[U]
end
Linear graph convolutions are linearized GNNs that come from the conditioning on features.

**Linear Graph Convolution (LGC)**

\[(1 - \alpha) (I + \alpha S + \alpha^2 S^2 + \ldots) X \beta \quad S = D^{-1/2} WD^{-1/2}\]

[Jia–Benson 21]

**Simplified Graph Convolution (SGC)**

\[\tilde{S}^K X \beta \quad \tilde{S} = (D + I)^{-1/2} (W + I)(D + I)^{-1/2}\]

[Wu+ 19]

**Graph Convolution Network (GCN)**

\[\sigma(\tilde{S}) \ldots \sigma(\tilde{S}X\Theta^{(1)}) \ldots \Theta^{(K)} \beta\]

[Kipf–Welling 17]
Our model helps us understand smoothing.

\[ N = V \Lambda V^T, \text{ feature } f = \sum_{i=1}^{n} c_i v_i \]

**LGC**  \[ f \rightarrow \sum_{i=1}^{n} \frac{1}{1 + \omega \lambda_i} c_i v_i \]

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

**SGC**  \[ f \rightarrow \sum_{i=1}^{n} (1 - \frac{d}{d+1})^{\lambda_i} c_i v_i \]

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

Our model helps us understand smoothing.

\[ f = \sum_{i=1}^{n} c_i v_i \rightarrow \sum_{i=1}^{n} \frac{1}{1 + \omega \lambda_i} c_i v_i \]

\[ f = \sum_{i=1}^{n} c_i v_i \rightarrow \sum_{i=1}^{n} (1 - \frac{d}{(d+1)\lambda_i})^K c_i v_i \]
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 $X_1$ and $|V_2| \times p$ matrix $X_2$ of node features (same features)
- Subset $L_1 \subset V$ of labeled nodes.
- Length-$|L_1|$ vector $y_{L_1}$ of outcomes on $L_1$.

**Problem output.**
- Length-$|V_2|$ vector $y$ 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.

\[ y_U^{LGC/RP} = [(I_n + \omega N)^{-1}X\beta]_U - (I + \omega N)^{-1}_{UU}(I + \omega N)_{UL}(y_L - [(I_n + \omega N)^{-1}X\beta]_L) \]
Label Propagation and Graph Neural Networks


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

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