Distributed Structural Estimation of Graph Edge-Type Weights from Noisy PageRank Orders

David Daniels, Eric Liu, Charles Zhang

CME 323
Not all edges are created equal

Example: AngelList ($2.9+ billion)

- Investors can follow a start-up (social link)
- Investors can invest in a start-up (economic link)
- What is the relative importance of a social link versus an economic link?
Goal

- This paper attempts to *recover* edge weights that are *revealed* via the propagation of actual influence along a network
  - “What is the edge–type weight vector that best describes a graph, assuming influence operates as if characterized by Edge–Type Weighted PageRank?”
Overview

- Edge–Type Weighted Graphs
- Weighted PageRank
- Structural Estimation
- Algorithm
  - (Inner) PageRank Iterations (for given weight vector)
  - (Outer) Search Strategy (for optimal weight vector)
- Experiments
Edge–Type Weighted Graphs

\[ G = (V, E(e, t), \omega) \in \mathcal{G}^w \]

- \( V \) is the set of vertices with \(|V| = n\)
- \( E \) is the set of edges with \(|E| = m\)
- Each edge \( e \) having a type attribute \( t \in \mathcal{T} = \{1, 2, \ldots, T\} \)
- \( \omega = (\omega_1, \ldots, \omega_T) \in \mathbb{R}_+^T \) is the weight vector
Weighted stochastic adjacency matrix \( A \in \mathbb{R}^{n \times n} \)

\[
A_{ji} = \frac{w_{ij}}{\sum_j w_{ij}} \quad \text{where} \quad w_{ij} = \omega_{t_{ij}} \quad \text{is the weight for edge} \quad e_{ij},
\]

if \( e_{ij} \in E \) (i.e. if \( v_i \to v_j \)), and \( w_{ij} = 0 \) if \( e_{ij} \notin E \)

Alternatively, we can write \( A \) as

\[
A = (\omega_1 B^{(1)} + \cdots + \omega_T B^{(T)}) C
\]

where \( C_{ii} = \sum_t \omega_t \left( \sum_j [B^{(t)}]_{ij} \right) \)
Weighted PageRank

Find \( r \in \mathbb{R}^n \) such that \( r = (1 - \delta)/n + \delta Ar \)

i.e \( \rightarrow \) Find the eigenvector corresponding to the eigenvalue \( \lambda = 1 \) for the matrix \( M = \delta A + \frac{1 - \delta}{n} \mathbf{1} \)

(Perron–Frobenius) on positive stochastic matrices

\( \lambda = 1 \) is the unique largest eigenvalue of \( M \nRightarrow \) Power Iterations
\[ \omega_{\text{opt}} = \arg \min_{\omega} h \left( \text{order} \left( PR(G(V,E,\omega)) \right), p^* \right) \]

- \[ p^* = \text{order} \left( PR(G(V,E,\omega^*)) + \mathcal{N}(0, \sigma^2 \epsilon I) \right) \]
- \[ h(p_1, p_2) = \| p_1 - p_2 \|_2 \]
Structural Estimation – Simulation

squared component errors for $w^* = (0.2, 0.3, 0.5)$
Structural Estimation – Simulation

squared component errors for $w^* = (0.6, 0.3, 0.1)$
Key results from simulations

- The optimal weight vector $\omega_{opt}$ that attains minimum of $f$ lies in close neighbourhood of true weight vector, $\omega^*$.

- Albeit intractability to find closed-form of derivative, the function $f$ is convex and smooth (and at least piecewise continuous/convex for higher dimensions).

- Weighted PageRanks perform better than unweighted PageRanks especially on graphs with high in-degree / low edge weights, low in-degree / high edge weights (see paper).
Structural Estimation – Simulation

\( \omega^* = (0.5714, 0.2857, 0.1429) \)

95% CI for \( \omega_1^* \): [0.5677, 0.5831]

95% CI for \( \omega_2^* \): [0.2797, 0.2899]

95% CI for \( \omega_3^* \): [0.1325, 0.1471]
Algorithm – PageRank Iteration

Local Machine

- **Power iteration on** $M$
  - $(2n - 1)n \sim O(n^2)$ per multiplication
  - Number of iterations:
    $$O \left( \frac{\log(1/\varepsilon) - cc_2/c_1}{\log(1/\lambda_2)} \right) \sim O \left( \frac{\log(1/\varepsilon)}{\log(1/\delta)} \right)$$

  because $M^k \nu = c_1 (r + \frac{c_2}{c_1} (\lambda_2)^k q_2 + \cdots + \frac{c_n}{c_1} (\lambda_n)^k q_n$
  and by Haveliwala (2003)’s bound on $|\lambda_2| \leq \delta$

- **Smart update**: $\nu^{(k)} = (1 - \delta)/n + \delta A \nu^{(k-1)}$
  - $2m - n \sim O(m)$ per update
Algorithm – PageRank Iteration

Distributed using Pregel Framework

Algorithm 1 PageRank

input: $G : \text{Graph}[V,E])$
while $err \geq \epsilon$ do
    for vertex $i$ do
        $R[i] = 0.15 + 0.85 \sum_{j \in \text{N}_\text{in}(i)} M[j]$
        $M[i] = R[i] / |\text{N}_\text{out}(i)|$
        Send $M[i]$ to all $\text{N}_\text{out}(i)$
    end for
    $err = |R - \text{previous} R|$
end while
Distributed using Pregel Framework, $B$ machines, with combiners

- Communication cost: $\mathcal{O}(\min(m, nB))$
- Reduce size for each key:
  - $\mathcal{O}(\min(\text{max indegrees}, B))$
  - Max in-degrees could be as bad as $\mathcal{O}(m)$
  - On average, it is $\mathcal{O}(m/n)$
- Number of supersteps: $\mathcal{O}(\log(1/\epsilon))$
Algorithm – Search Strategy

- Grid search
  \[ \mathcal{O}\left((1/\alpha)^T m \log(1/\epsilon)\right) \]

- Numerical gradient descent
  \[
  \omega^{(l+1)} = \omega^{(l)} - \gamma \nabla f(\omega^{(l)})
  \]
  \[
  \frac{\partial f}{\partial \omega_t} (\omega^{(l)}) = \frac{f(\omega^{(l)} + \alpha e_t - \alpha e_T) - f(\omega^{(l)})}{\alpha}
  \]
  \[ \mathcal{O}(ST(m \log(1/\epsilon) + n \log n)L) \]
Experiments

- Accuracy?

N.B.: Experiments use a slightly different minimization objective – squared difference in PageRank scores rather than squared difference in PageRank order – because our laptop Spark setup (4 cores, 4 GB memory) was able to process the former metric, but not the latter, in a reasonable time for graphs of a size worth distributing.
### Experiments

#### Accuracy?

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Edges</th>
<th># Types</th>
<th>Recovered Weights</th>
<th>True Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>20%</td>
<td>2</td>
<td>.332, .668</td>
<td>.33, .67</td>
</tr>
<tr>
<td>600</td>
<td>20%</td>
<td>4</td>
<td>.098, .199, .3, .4</td>
<td>.1, .2, .3, .4</td>
</tr>
<tr>
<td>600</td>
<td>20%</td>
<td>2</td>
<td>.331, .669</td>
<td>.33 + \epsilon, .67 + \epsilon, \epsilon \sim \mathcal{N}(0, (\frac{1}{\sum w_i})^2)</td>
</tr>
<tr>
<td>2000</td>
<td>500</td>
<td>3</td>
<td>.165, .332, .503</td>
<td>.166, .333, .5</td>
</tr>
</tbody>
</table>
Experiments

- Runtime?
Erdos-Renyi random graphs, 600 Nodes
Can recover edge–type weights accurately

Next steps

- Dynamically changing PageRank tolerance
- Look for direction in unit ball with small $\lambda_2$
Thank You!