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

David Daniels, Eric Liu, Charles Zhang

CME 323

Introduction

- 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 $t \in \mathcal{T} = \{1, 2, \dots, T\}$
- $\omega = (\omega_1, \dots, \omega_T) \in \mathbb{R}_+^{\mathrm{T}}$ is the weight vector

Edge-Type Weighted Graphs

Weighted stochastic adjacency matrix A$\in \mathbb{R}^{n\times n}$

$A_{ji} = \frac{w_{ij}}{\sum_j w_{ij}}$ where $w_{ij} = \omega_{t_{ij}}$ is the weight for edge e_{ij}, if $e_{ij} \in E$ (i.e. if $v_i \rightarrow v_j$), and $w_{ij} = 0$ if $e_{ij} \notin E$

Alternatively, we can write A as

$$A = \left(\omega_1 B^{(1)} + \cdots + \omega_T B^{(T)}\right)C$$

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

Weighted PageRank

- Find $r \in \mathbb{R}^n$ such that $r = (1-\delta)/n \ + \delta A r$
- i.e -> Find the eigenvector corresponding to the eigenvalue $\lambda = 1$ for the matrix $M = \delta A + \frac{1-\delta}{n}\mathbb{1}$
- (Perron–Frobenius) on positive stochastic matrices

  $\lambda = 1$ is the unique largest eigenvalue of M

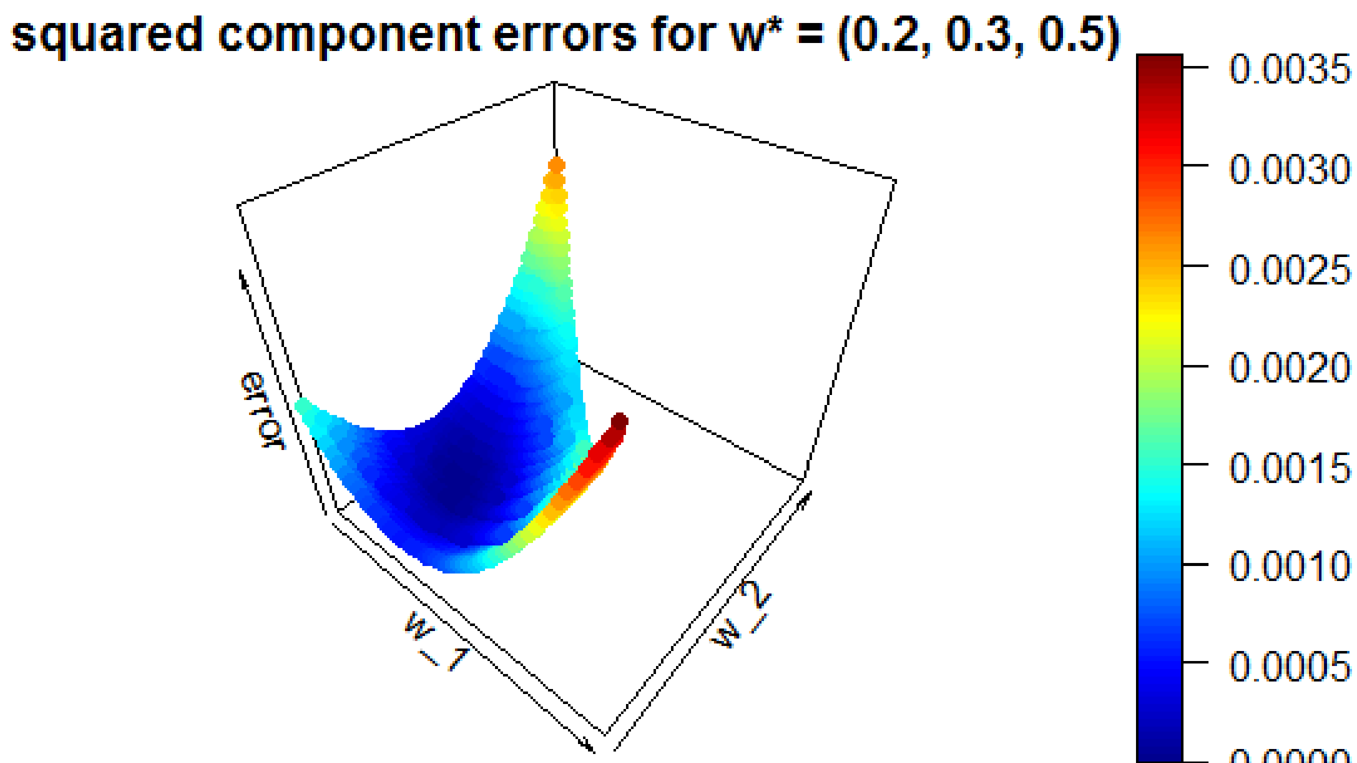
  $=>$ Power Iterations

Structural Estimation

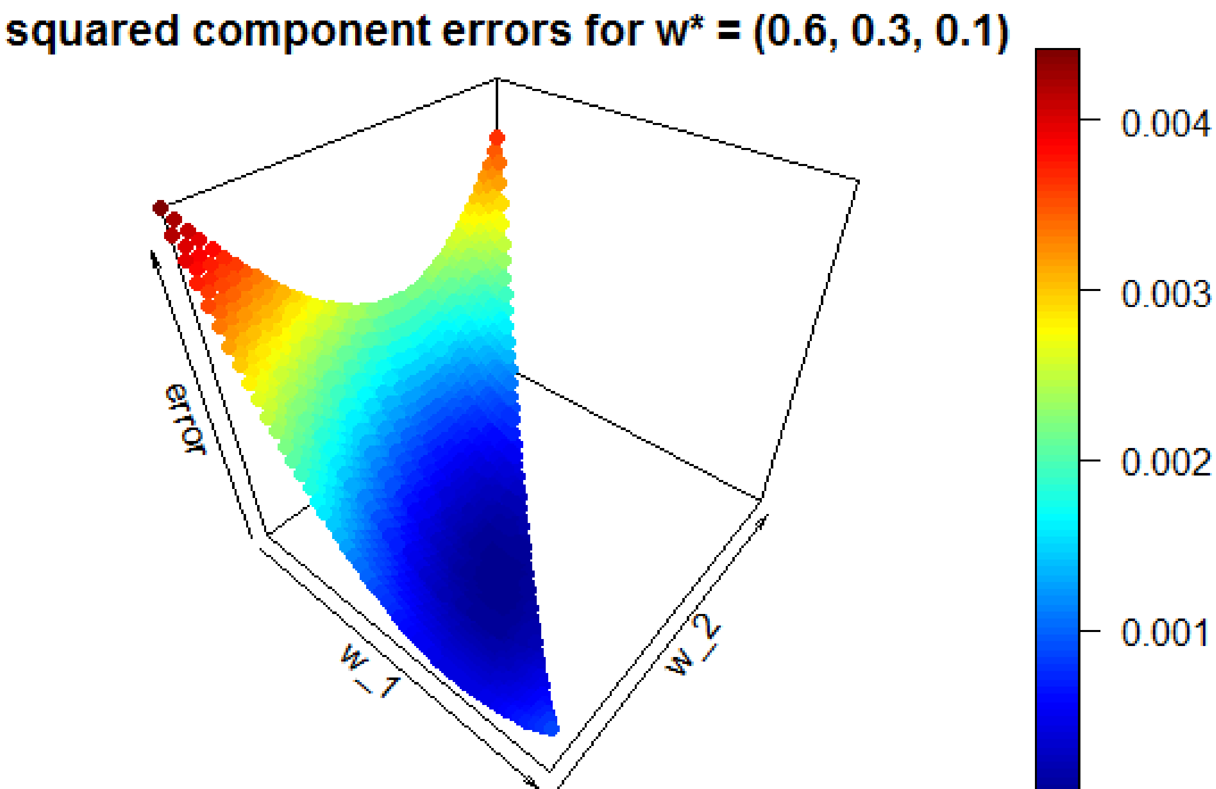
$$\omega_{\text{opt}} = \underset{\omega}{\operatorname{argmin}}\ h\left(order\left(PR\big(G(V,E,\omega)\big)\right), p^*\right)$$

- $p^* = order(PR\big(G(V,E,\omega^*)\big) + \mathcal{N}(0, \sigma_\epsilon^2 I))$
- $h(p_1, p_2) = \|p_1 - p_2\|_2$

Structural Estimation – Simulation



Structural Estimation – Simulation

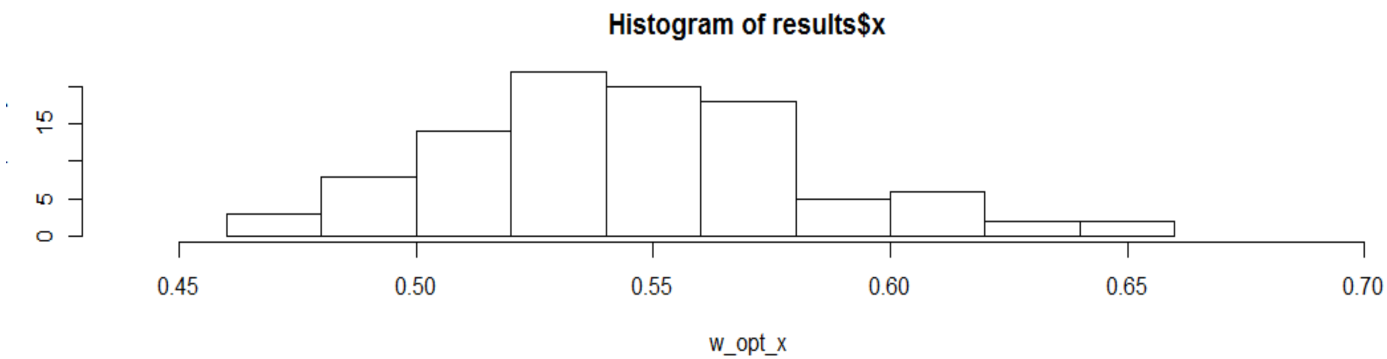


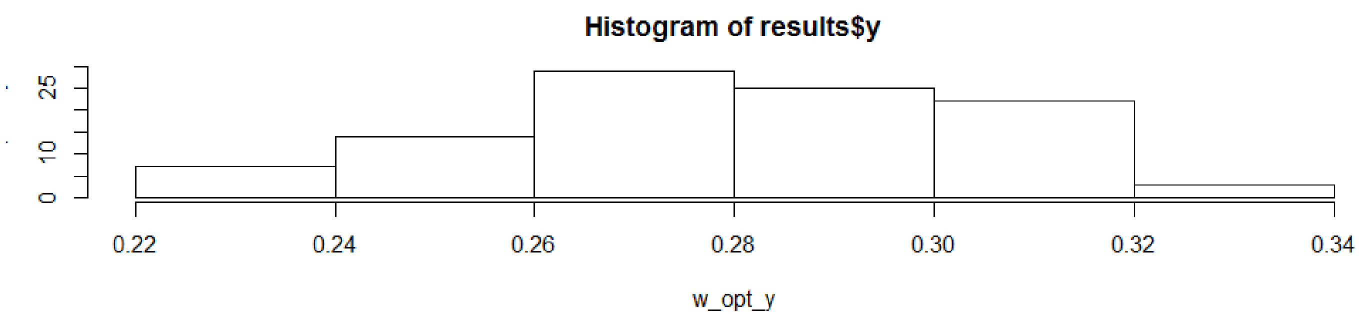
Structural Estimation – Simulation

Key results from simulations

- The optimal weight vector ω_{opt} that attains minimum of f lies in close neighbourhood of true weight vector, ω^*.
- 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 ω_1^* : [0.5677, 0.5831]
95% CI for ω_2^* : [0.2797, 0.2899]
95% CI for ω_3^* : [0.1325, 0.1471]

Algorithm – PageRank Iteration

Local Machine

- Power iteration on M
  - $(2n-1)n \sim \mathcal{O}(n^2)$ per multiplication
  - Number of iterations:

$$\mathcal{O}\left(\frac{\log(1/\epsilon) - cc_2/c_1}{\log(1/\lambda_2)}\right) \sim \mathcal{O}\left(\frac{\log(1/\epsilon)}{\log(1/\delta)}\right)$$

because $M^k v = c_1(r + \frac{c_2}{c_1}(\lambda_2)^k q_2 + \cdots + \frac{c_n}{c_1}(\lambda_n)^k q_n$

and by Haveliwala (2003)'s bound on $|\lambda_2| \leq \delta$

- Smart update: $v^{(k)} = (1-\delta)/n \ + \delta A v^{(k-1)}$
  - $2m - n \sim \mathcal{O}(m)$ per update

Algorithm – PageRank Iteration

Distributed using Pregel Framework

Algorithm 1 PageRank

---

input: $G : Graph[V, E])$
while $err \geq \epsilon$ **do**
    for vertex i **do**
$$R[i] = 0.15 + 0.85 \sum_{j \in N_{\text{in}}(i)} M[j]$$
$$M[i] = R[i]/|N_{\text{out}}|$$
        Send $M[i]$ to all $N_{\text{out}}(i)$
    end for
$$err = |R - previousR|$$
end while

Algorithm – PageRank Iteration

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

$$\mathcal{O}\big((1/\alpha)^T m \log(1/\epsilon)\big)$$

- Numerical gradient descent

$$\omega^{(l+1)} = \omega^{(l)} - \gamma \, \nabla f\big(\omega^{(l)}\big)$$

$$\frac{\partial f}{\partial \omega_t}\big(\omega^{(l)}\big) = \frac{f\big(\omega^{(l)} + \alpha e_t - \alpha e_T\big) - f\big(\omega^{(l)}\big)}{\alpha}$$

$$\mathcal{O}(ST(m \log(1/\epsilon) + n \log n)L)$$

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.

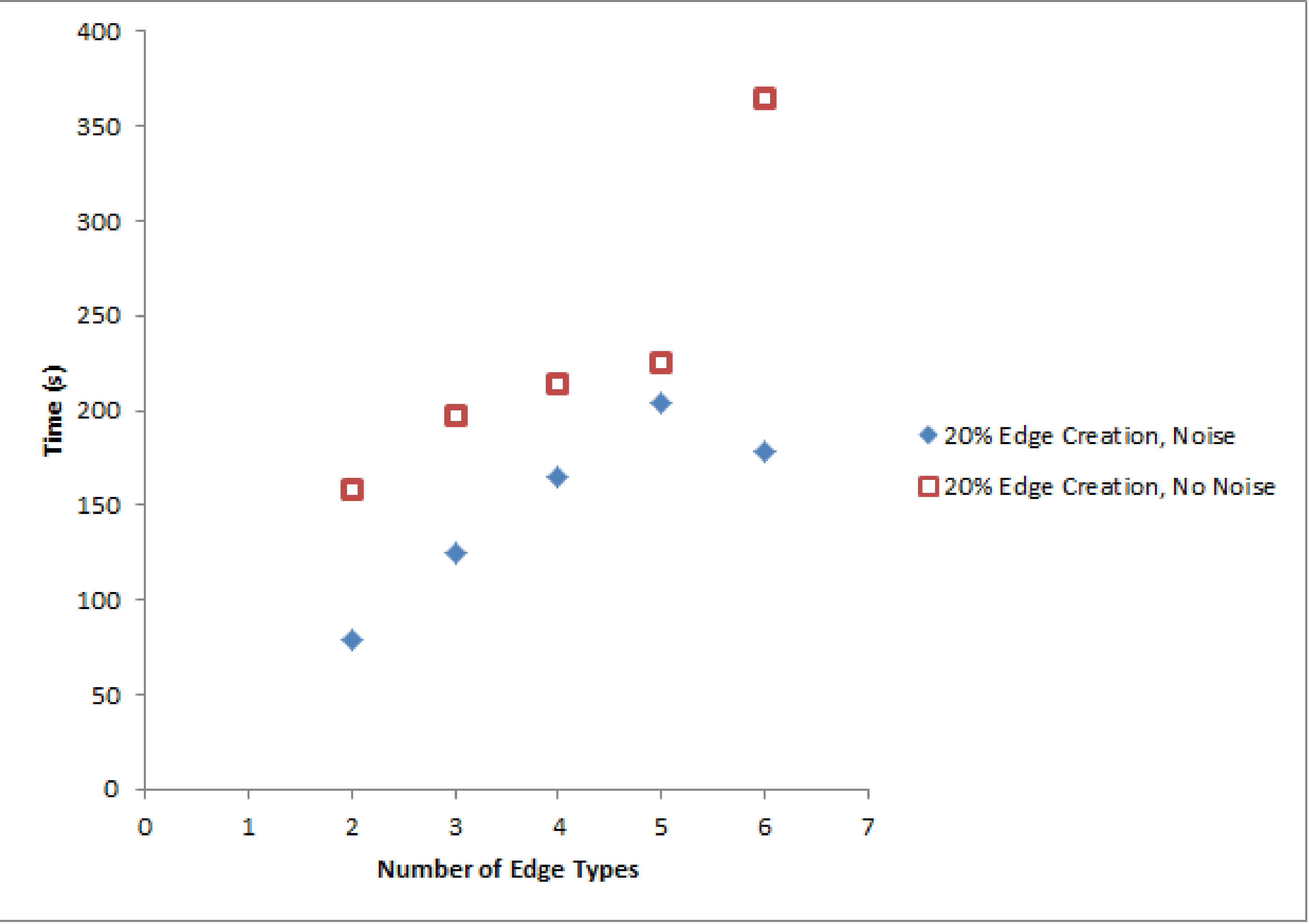
Experiments

- Accuracy?

Nodes	Edges	# Types	Recovered Weights	True Weights
600	20%	2	.332, .668	.33, .67
600	20%	4	.098, .199, .3, .4	.1, .2, .3, .4
600	20%	2	.331, .669	$.33+\epsilon, .67+\epsilon,$ $\epsilon \sim \mathcal{N}\left(0, \left(\frac{.1}{\sum w_i}\right)^2\right)$
2000	500	3	.165, .332, .503	.166, .333, .5

Experiments

- Runtime?



Erdos-Renyi random graphs, 600 Nodes

Conclusion

- Can recover edge-type weights accurately
- Next steps
  - Dynamically changing PageRank tolerance
  - Look for direction in unit ball with small λ_2

Thank You!