Almost-Matching-Exactly for Treatment Effect Estimation under Network Interference

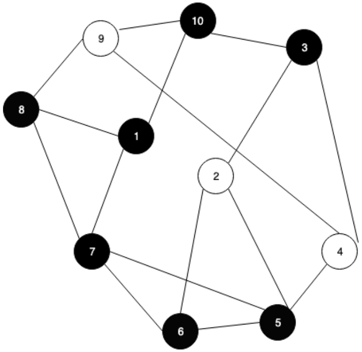
Usaid Awan, Marco Morucci, Vittorio Orlandi

Setting - Causal Inference

- We have $i = 1, \ldots, n$ experimental units
- Treatment $t_i \in \{0, 1\}$ with $\mathbf{t} \in \{0, 1\}^n$ is a binary vector with the treatment level of every unit.
- Potential outcomes $Y_i(t_i, \mathbf{t})$ are random variables and depend on both treatment of unit i (1st argument), and treatment of **all other units** (2nd argument).
- Observed treatment $\mathbf{T} \in \{0, 1\}^n$ is assigned **uniformly at random**.
- Observed outcome: $Y_i = T_i Y(1, \mathbf{T}) + (1 - T_i) Y_i(0, \mathbf{T})$
  - Since treatment is randomized: $\mathbb{E}[Y_i | \mathbf{T} = \mathbf{t}, T_i = t] = \mathbb{E}[Y_i(t, \mathbf{T})]$ (Ignorability).
- Units are connected in a network, G, in which unit i's **treated neighborhood subgraph** is $G^{\mathbf{t}}_{\mathcal{N}_i}$.

The Problem: No SUTVA!

Usually we assume SUTVA: that *units' treatments don't influence other units' outcomes*, but we can't do that here because our units are connected in a network:

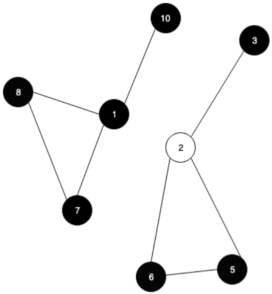


It could be that the treatment assigned to j influences the outcome of i through their connection in the network.

Similar Graphs Carry Similar Interference

Idea

What if the amount of interference experienced by a unit depended on the *shape* of its treated neighborhood subgraph?



Then, in expectation, two units with the same treated neighborhood graph will respond similarly to the treatment.
We can use this idea to do matching to reduce interference.

Assumptions

1. Outcome model: $Y_i = \alpha + t_i\beta_i + f_i(G^{\mathbf{t}}_{\mathcal{N}_i}) + \epsilon_i$
   - f is some interference function dependent on $G^{\mathbf{t}}_{\mathcal{N}_i}$, the **treated neighborhood subgraph** of unit i.
2. $\mathbb{E}[\epsilon_i | T_i] = \mathbb{E}[\epsilon_i] = 0$
   - Ignorability
3. If $G^{\mathbf{t}}_{\mathcal{N}_i} \simeq G^{\mathbf{t}}_{\mathcal{N}_j}$ then $f_i(G^{\mathbf{t}}_{\mathcal{N}_i}) = f_j(G^{\mathbf{t}}_{\mathcal{N}_j}) = f(G^{\mathbf{t}}_{\mathcal{N}_i})$.
   - Two units with isomorphic neighborhood sugraphs experience the same interference.
   - Together with (1), this assumption encodes a version of SANIA (Airoldi and Sussman, 2018) conditional on unit's neighborhood subgraphs.

Matching

- Under these assumptions we can match units to recover the effect we are interested in (τ_i)
- For a treated unit i, find a control unit j, such that $G^{\mathbf{t}}_{\mathcal{N}_i} \simeq G^{\mathbf{t}}_{\mathcal{N}_j}$
- Subtract Y_j from Y_i to recover τ_i in expectation

$$\mathbb{E}[Y_i - Y_j] = \mathbb{E}[\tau_i + f(G^{\mathbf{t}}_{\mathcal{N}_i}) - f(G^{\mathbf{t}}_{\mathcal{N}_j}) + \epsilon_i + \epsilon_j] = \tau_i$$

If no unit has exactly $G^{\mathbf{t}}_{\mathcal{N}_j} \simeq G^{\mathbf{t}}_{\mathcal{N}_i}$ we can find a unit that has a neighborhood graph that is not quite equal to $G^{\mathbf{t}}_{\mathcal{N}_i}$ but is similar enough.

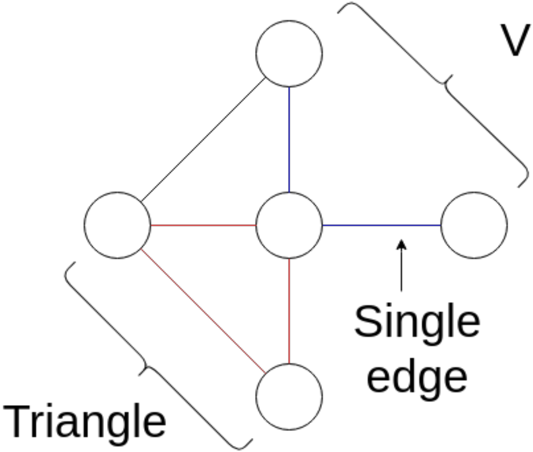
Problem

How do we represent similarity between neighborhood graphs?

Subgraph Counts

- We'll say that neighborhood graphs are similar if they contain similar counts of subgraphs
- In fact, if the counts of subgraphs are the exact same, then the neighborhood graphs must be isomorphic.
- We match together units that have the most similar subgraphs
- If no unit matches exactly to another on all subgraph counts, then we must choose which subgraphs we want to match exactly on and which ones we can afford to ignore.

Subgraph Counts



This graph has 2 triangles, 6Vs and 4 single edges.

Problem #2

How do we choose which subgraphs we should use to represent the treated neighborhood graphs of our units?

We need a matching method that selects which subgraphs to match on

1. Enumerate (up to isomorphism) all p subgraphs $S_1, \ldots, S_p$ seen across all the $\mathcal{N}_i$, $i = 1, \ldots, n$
2. For each unit i, define the p-dimensional vector $S(G_{\mathcal{N}_i^{\mathbf{t}}})$ as the vector whose j'th entry is the number of S_j in $\mathcal{N}_i$
3. These are likely a lot (maximum on the order of $|\mathcal{N}_i|^2$) and it's unlikely that many units will have identical counts
4. The FLAME matching algorithm will automatically select the best subgraphs to match on

FLAME: An Overview

- FLAME (Fast Large-Scale Almost Matching Exactly) is a method for creating interpretable matches between units with discrete covariates that performs variable selection while matching.
  1. Match units exactly on as many covariates as possible
  2. Drop a covariate
  3. Repeat
- At each step, drop the covariate maximizing match quality:

$$\mathtt{MQ} = C \cdot \mathtt{BF} - \mathtt{PE}$$

- $\mathtt{BF}$ = prop. controls matched + prop. treated matched
- $\mathtt{PE}$ = prediction error achieved by remaining covariates
- Tradeoff between making matches and accurate prediction

A Small Change to FLAME

We know from our theoretical setup that network statistics should do two things well:

1. Predict the **outcomes**
2. Predict the **network**

To measure how well the our network statistics (subgraphs) are predicting the network, we model the edges of the network as independent conditional on the observed statistics

$$E_{ij}|x_i, x_j \overset{iid}{\sim} \text{Bern}(\text{logit}(\beta_1' x_i + \beta_2' x_j))$$

and consider the AIC of the resulting model, $\text{AIC}_{\text{network}}$.

The Modified PE Function

To ensure FLAME strikes a balance between predicting both the **outcomes** and the **network**, we modify the PE function:

$$\mathsf{PE} = \sum_{t=0}^{1} \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - f(S(G^{\mathbf{t}}_{\mathcal{N}_i})))^2 - \underbrace{D \cdot \mathrm{AIC}_{\mathrm{network}}}_{\text{new component}}$$

Simulation Setup

- Recall the outcome model: $Y_i = \alpha + \beta_i t_i + f(G_{\mathcal{N}_i^t}) + \epsilon_i$
- The graph G is generated according to Erdos-Renyi or Stochastic Block models
- We consider various forms of f based off different features:
    - d_i: the degree of unit i
    - Δ_i: the number of triangles in $\mathcal{N}_i$
    - $\dagger_i^k$: the number of units in $\mathcal{N}_i$ with degree $\geq k$
    - $\bigstar_i^k$: the number of k-stars in $\mathcal{N}_i$
    - Betweenness$_i$: the vertex betweenness of unit i
    - Closeness$_i$: the closeness centrality of unit i

Estimators

- True: nearest neighbor (NN) on true interference
- Naive: naive difference in means
- Eigen All: NN on eigenvalues of adjacency matrix A
- Eigen All: NN on largest eigenvalue of A
- Stratified Naive: Stratified degree estimator
- SANIA: MIVLUE under SANIA
- FLAME: Our approach

(OLD) Simulation Results

Test 1: Simple Interference

Interference:

$$f_i(\mathbf{z}) = \gamma d_i(\mathbf{z})$$

Outcomes:

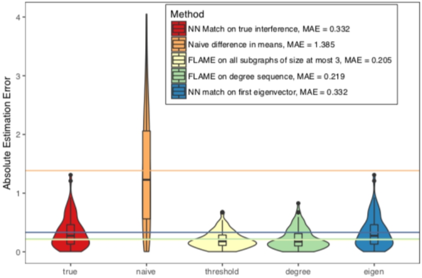
$$Y_i = \alpha_i + \beta_i z_i + \gamma d_i(\mathbf{z})$$

Parameters:

$$N = 100,\ Nsim = 100,\ \alpha_i \sim \mathcal{N}(0,1),\ \beta_i \sim \mathcal{N}(5,1),\ \gamma = 4,\ G \sim ER(0.05),\ Z_i \sim Ber(0.5)$$

3

Results:



(OLD) Simulation Results

Test 3a: Even More complex Interference (denser graph)

Interference:

$$f_i(\mathbf{z}) = d_i(\mathbf{z})(\delta\Delta_i(\mathbf{z}) + \lambda V_i(\mathbf{z}) + \eta\bigstar_i^4(\mathbf{z}))$$

Outcomes:

$$Y_i = \alpha_i + \beta_i z_i + f_i(\mathbf{z}).$$

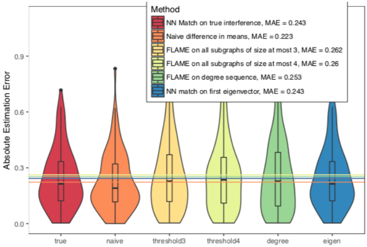
Parameters:

$$N = 100,\ Nsim = 100,\ \alpha_i \sim \mathcal{N}(0,1),\ \beta_i \sim \mathcal{N}(5,1),\ \delta = 2,\ \lambda = 0.5,\ \eta = 4; G \sim ER(0.1),\ Z_i \sim Ber(0.5)$$

.

Results:

6



(OLD) Simulation Results

Test 5b: Interference that is not related to subgraph counts (denser graph)

Interference:

$$f_i(\mathbf{z}) = \delta * Betweenness_i(\mathbf{z}) + \lambda * Closeness_i(\mathbf{z})$$

Outcomes:

$$Y_i = \alpha_i + \beta_i z_i + f_i(\mathbf{z}).$$

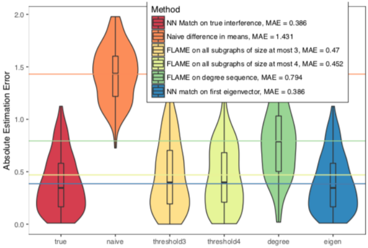
Parameters:

$$N = 100,\ Nsim = 100,\ \alpha_i \sim \mathcal{N}(0,1),\ \beta_i \sim \mathcal{N}(5,1),\ \delta = 40,\ \lambda = 20,\ G \sim ER(0.1),\ Z_i \sim Ber(0.5)$$

.

Results:

10



Plan: We Want to Hit the October 8 AISTATS Deadline

1. We need to implement the revised PE function
2. We need to redo the simulations
3. More theory? Statements on how well the subgraph count can "encode" a given graph would be nice
4. Find a good application
5. Write!