Reconstructing randomized graphs

Niko Vuokko

Evimaria Terzi

Objective

- Reconstruct a graph and/or feature set that has been randomized by a known algorithm

- Why
  - Attack privacy
  - Lossy/Noisy information
  - Data analysis – (Congressional voting?)

Problem Definition

- Given an observed graph (G') and feature set (F') try to rebuild the original G and F.
- Possible observed pairs:
  - G' = G and F' != F (G is not randomized, F is)
  - G' != G and F' = F (F is not randomized, G is)
  - G' != G and F' != F (both G and F are randomized)
- All possible observations can be solved in Polynomial time!!!!

Preliminaries

- Graph (G) represented by an adjacency matrix
  - seen this before 1 represents edge, 0 no edge
  - g_{ij} represents edge in G
- Feature set (F) represented by matrix of 0 – 1 values
  - row represents a node, column represents a feature
  - if graph has n nodes and each node has k features there are nk entries in matrix
  - $\mathbf{f}_i$ is a feature vector in F, f_{ilk} represents feature

Preliminaries

- Relationship between nodes and features?
  - From Plato, “Friends have all things in common.”
  - Key assumption, relates features and nodes

- Edge exists if two nodes have features in common
  - Use a similarity function between feature vectors

Preliminaries

- Similarity Function types used in the paper:
  - Dot Product (DP): $\mathrm{sim}(\boldsymbol{f_i}, \boldsymbol{f_j}) = \sum_{l=1}^{k} f_{il} \cdot f_{jl}$
  - Hamming (H): $\mathrm{sim}(\boldsymbol{f_i}, \boldsymbol{f_j}) = \sum_{l=1}^{k} 1 - |f_{il} - f_{jl}|$
- More similar features, more probable an edge

$$Pr(g_{ij} = 1 | \boldsymbol{f_i}, \boldsymbol{f_j}) = \frac{1}{Z} \mathrm{e}^{\alpha\,\mathrm{sim}(\boldsymbol{f_i}, \boldsymbol{f_j})}$$

$$Pr(g_{ij} = 0 | \boldsymbol{f_i}, \boldsymbol{f_j}) = \frac{1}{Z} \mathrm{e}^{\alpha(1 - \mathrm{sim}(\boldsymbol{f_i}, \boldsymbol{f_j}))}$$

Preliminaries

- Assume edges are independent and features define edges

$$Pr(G,F) = Pr(G|F) = \prod_{i<j} Pr(g_{ij}|\boldsymbol{f_i},\boldsymbol{f_j})$$

- Randomization – From matrix X produce X'
  - Key probabilities needed:
    - Pr(x'=0|x=0) and Pr(x'=1|x=0)
    - Pr(x'=0|x=1) and Pr(x'=1|x=1)
  - Must be able to calculate Pr(X|X')

$$Pr(X|X') = \frac{Pr(X'|X)\,Pr(X)}{Pr(X')} \propto Pr(X'|X) = \prod_{i,j} Pr(x'_{ij}|x_{ij})$$

Reconstruction

- Uses a maximum likelihood approach
  - Given observation which actual is most probable?

- Three types of problems:
  - G – reconstruction (when F not altered)
  - F – reconstruction (when G not altered)
  - GF – reconstruction (when both altered)

- In short want to find Pr(G,F | G',F')

Reconstruction

- Instead of maximizing Pr(G, F | G', F'), minimize the -log(Pr(G,F | G', F'))
- How?

$$Pr(G,F|G',F') = \frac{Pr(G',F'|G,F)\,Pr(G,F)}{Pr(G',F')}$$

$$Pr(G,F|G',F') \propto Pr(G',F'|G,F)\,Pr(G,F) = Pr(G'|G)\,Pr(F'|F)\,Pr(G,F)$$

Energy Function for minimization -

$$E(G,F) = -\log Pr(G,F|G'F') = -\log Pr(G'|G) - \log Pr(F'|F) - \log Pr(G,F)$$

Reconstruction

- GF – reconstruction energy function:

$$E(G,F) = -\log Pr(G,F|G'F') = -\log Pr(G'|G) - \log Pr(F'|F) - \log Pr(G,F)$$

- G – reconstruction energy function:

$$E(G) = -\log Pr(G'|G) - \log Pr(G,F) = \sum_{i<j} (-\log Pr(g'_{ij}|g_{ij}) - logPr(g_{ij}|\boldsymbol{f}'_i, \boldsymbol{f}'_j))$$

- F – reconstruction energy function:

$$E(F) = -\log Pr(F'|F) - \log Pr(G,F) = \sum_{i=1}^{n} \sum_{l=1}^{k} (-logPr(f'_{il}|f_{il}) - \log Pr(g'_{ij}|\boldsymbol{f_i}, \boldsymbol{f_j}))$$

Algorithms

- G – reconstruction, rebuild edges
  - For every two nodes, calculate the energy of an edge using E(G).
  - If E(edge) < E(no edge) then add an edge else no edge, remember we are trying to minimize energy
  - Optimal algorithm but not guaranteed to rebuild the original graph
  - Running time: $O(T_s n^2)$

Algorithms

- F – reconstruction, label feature values 0-1
  - Optimal algorithm from computer vision
    - Uses Min cut algorithm in a unique way (very cool!!)
    - Polynomial solution but expensive for computation time and space requirements
  - Naïve suboptimal algorithm
    - Performs labeling in greedy fashion
    - Makes assignment that best minimizes E in that move
    - Iteration based...

Algorithms

- Optimal F – reconstruction
  - Intuition, assign labels (1/0) to nk features
  - First rewrite E(F):

$$E(F)=\sum_{i=1}^{n}\sum_{l=1}^{k}(-logPr(f'_{il}|f_{il})-\log Pr(g'_{ij}|\boldsymbol{f_i},\boldsymbol{f_j}))=\sum_{i=1}^{n}\sum_{l=1}^{k}(\gamma(f_{il})-\delta(f_i,f_j))$$

  - Next build a flow-graph where each f_{il} is a node v_{ij} and add two terminals **s**, **t**.

Algorithms

After adding edges run
Min Cut Algorithm
and label
S = 0, T = 1

If: $\gamma(f_{il}=1) > \gamma(f_{il}=0)$

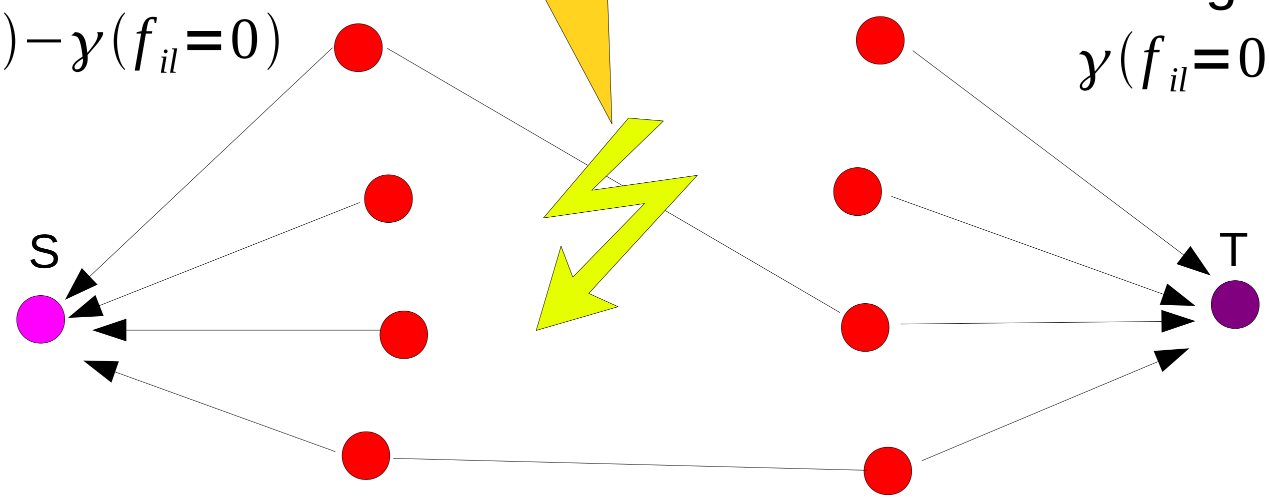
With weight:

$$\gamma(f_{il}=1) - \gamma(f_{il}=0)$$

If: $\gamma(f_{il}=0) > \gamma(f_{il}=1)$

With weight:

$$\gamma(f_{il}=0) - \gamma(f_{il}=1)$$



is an f_{il}

If edge in G,
Edge with weight:

$$(\delta(0,0) + \delta(1,1) - \delta(0,1) - \delta(1,0))/2$$

Algorithms

Connection to terminals

- To **s** if $\gamma(f_{il}=1)>\gamma(f_{il}=0)$
  - with weight $\gamma(f_{il}=1)-\gamma(f_{il}=0)$
- To **t** if $\gamma(f_{il}=0)>\gamma(f_{il}=1)$
  - with weight $\gamma(f_{il}=0)-\gamma(f_{il}=1)$

Connection to other nodes

- When edge in G, add edge to flow graph
  - with weight $(\delta(0,0)+\delta(1,1)-\delta(0,1)-\delta(1,0))/2$

Algorithms

- Perform min cut of flow graph
  - Nodes attached to **s** are labeled 0
  - Nodes attached to **t** are labeled 1

- Pretty Cool? Huh?

- More Info V. Komogorov, R. Zabih *What Energy Functions Can Be Minimized via Graph Cuts?*

Algorithms

- GF - reconstruction
  - Similar to F – reconstruction
  - Naïve Algorithm
    - Assigns values that minimize energy based on current move; greedy, suboptimal
  - Optimal Algorithm
    - Same as F-reconstruction but with a few modifications

Algorithms

- Optimal GF - reconstruction
  - Intuition – assign labels (1/0) to nk feature nodes, nC2 edges, and k(nC2) triples representing edge feature relationship
  - Requires manipulations of E(G,F), not presented in paper but explained as 'simple'
  - Limited to DP similarity functions only
    - Restriction based on behavior of energy function

Algorithms

- Manipulations result in three new edge evaluations
  - For g_{ij}: $\sigma_g(g_{ij}) = \alpha k g_{ij} - \log Pr(g'_{ij} | g_{ij})$
  - For f_{il}: $\sigma_f(f_{il}) = -\log Pr(f'_{il} | f_{il})$
  - For g_{ij}, f_{il}, f_{jl} tuples: $\sigma(g_{ij}, f_{il}, f_{jl}) = \alpha(1 - 2\mathrm{g}_{ij})\,\mathrm{sim}(f_{il}, f_{jl})$
- Evaluations of edges occurs identically to the optimal F – reconstruction algorithm

Algorithms

- Computational speedups
  - Optimal F and GF and naïve are very expensive
  - Solution is to divide the input space up, solve each subdivision, and aggregate results
  - Proposed algorithm uses a BFS tree

Experimental Results

- Tested algorithms against 3 datasets
  - A synthetic built dataset with n=200 and k=20
    - Controlled edge probability for 557 edges
  - DBLP dataset of author publications n=4981, k=19, and 20670 edges
  - Terror dataset
    - Nodes are attacks, features are attack characteristics, and edge exists if attack occurred at same location
    - N=645, k=94, and edges=3172

Experimental Results

- G – reconstruction
  - Used DBLP dataset
  - Tested against data subjected to increasing randomization amounts
  - Optimal algorithm performs pretty well in the presence of noisy data.
    - Error rate stabilizes at .625 as randomization increases

Experimental Results

- F – reconstruction
  - Used synthetic dataset
  - Tested against data subjected to increasing randomization amounts
  - Bounded naïve algorithm iterations to “clock time” of optimal solution
  - For small amounts of randomization naïve and optimal are close in error rate, for larger levels naïve performs worse than optimal

Experimental Results

- GF – reconstruction
  - Used all 3 datasets, only reported findings of Terror
  - Tested against data subjected to increasing randomization amounts
  - Various results:
    - For DP sim: OptBoth with split better than naïve both and split naïve both
    - For H sim: naïve both better than naïve both with split

Experimental Results

- DP sim function “lures” the reconstruction methods to fill up entries with 1s. Can corrected for by proper tuning of edge probability function

- Also noted that the objective function might result in (G,F) with high likelihood but low structural similarity to the data

Project Details

- In General
  - Rebuilding friendships from group information on Facebook
  - Are groups enough on Facebook to define friendships? Do we need more?
  - Challenges are getting a comprehensive dataset with enough group information for analysis
  - Thoughts?

The End

- Questions?

- Thank you