Latent Channel Networks

CASIS Workshop 2019

Clifford Anderson-Bergman

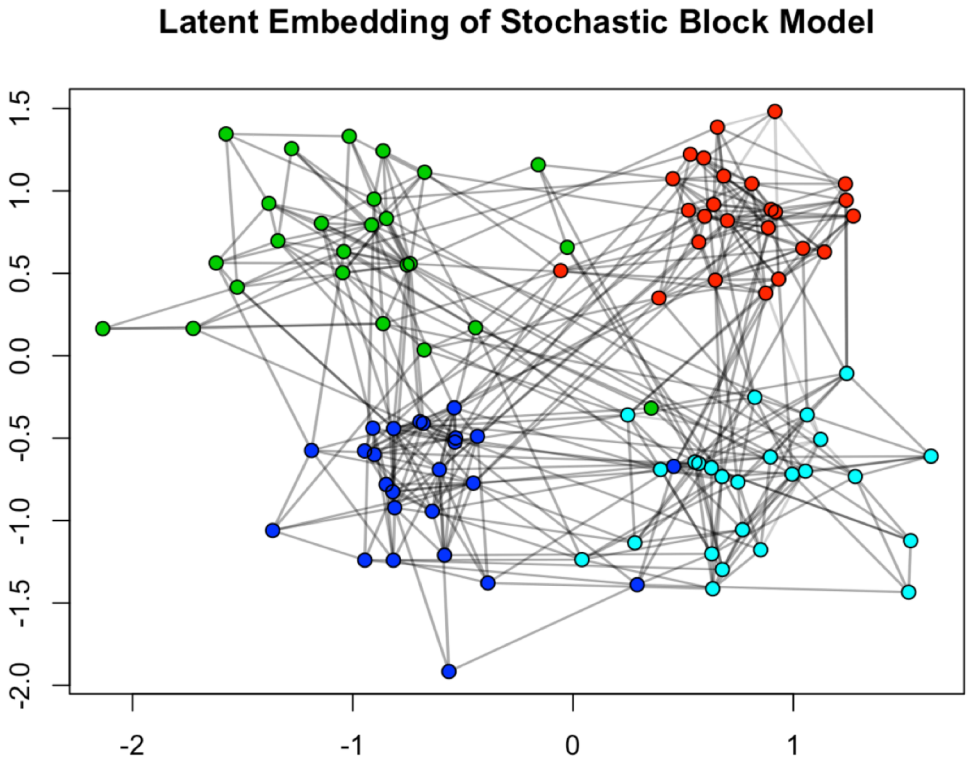
LLNL-PRES-773981

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Background: Latent Euclidean Embedding

- Probabilistic graph model
- Nodes are embedded in low-dimensional space
- Edge probability between two nodes inversely related to distance in Euclidean space
- Expanded to allow metadata affect edge probability



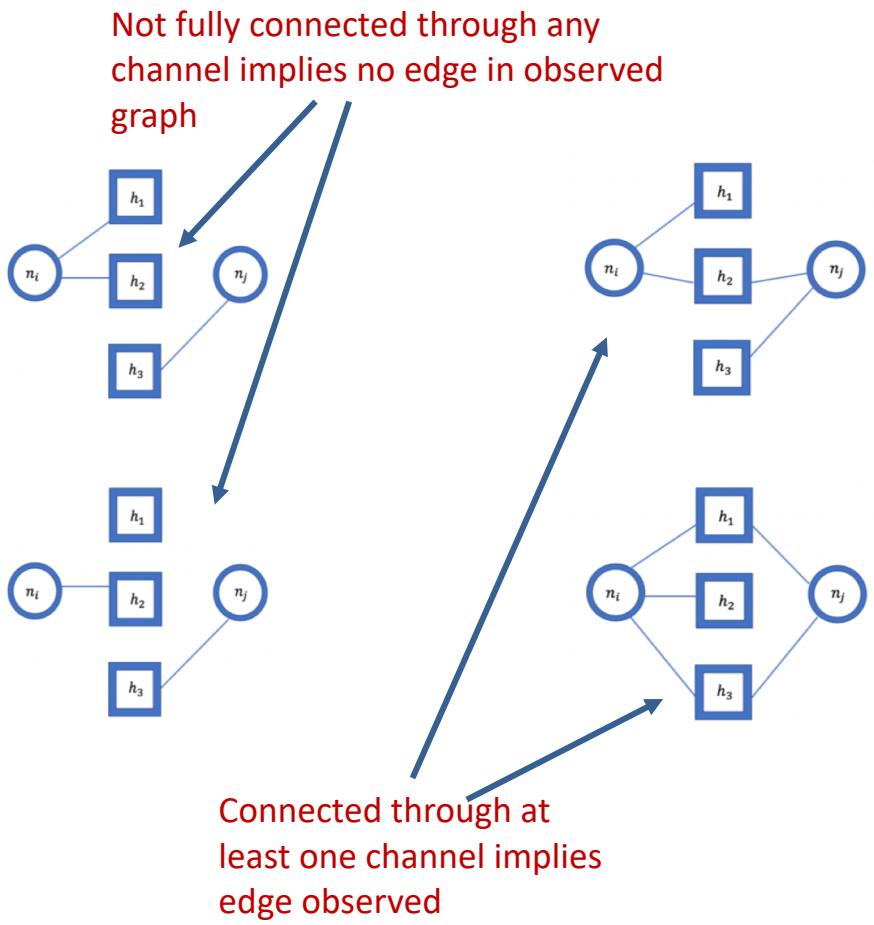
- P. D. Hoff, A. E. Raftery, and M. S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, 2002.

Social Networks

- We argue modern social networks are **not** well described by Latent Euclidean Space
  - In Euclidean Space, being far away in *any* dimension implies large distance
  - With modern communications, large numbers of potentially overlapping communities
    - Work
    - Hobbies
    - Politics
  - Having overlapping interest in *at least* one of these communities can imply high edge probability
  - Modern technology dropping physical distance constraints
- We would prefer model in which allows for high edge probability if similar in *one or more* dimensions
- May also like to be able to determine *which* dimensions nodes are similar on

Latent Channel Model: Heuristic Explanation

- Nodes of a graph can connect through *latent channels*
- We observe an edge in the graph if two nodes are connected through *at least* one channel
- Probability two nodes will share an edge through a given hub is a product of each node's frequency of use of that channel
- *Model easily accounts for high degree nodes: these are nodes that frequently use potentially many channels!*



Latent Channel Model: Formal Model

- e_{ikj}: Hidden edge from node i through channel k to node j
- p_{ik}: Marginal probability e_{ikj} = 1
- $p_{ik}\ p_{jk}$: Probability nodes i and j connect through channel k
- $1 - \prod_{k=1}^{K}(1 - p_{ik}p_{jk})$ Probability nodes i and j connect through at least one channel

- Full log likelihood function:

$$L(G|p) = \sum_{i=2}^{n}\sum_{j=1}^{i} e_{ij}\log\left(1 - \prod_{i=1}^{K}(1 - p_{ik}p_{jk})\right) + (1 - e_{ij})\log\left(\prod_{i=1}^{K}(1 - p_{ik}p_{jk})\right)$$

Interpreting Model

- Frequency of use of channel k by node i: p_{ik}
- If node i connected to node j, probability it is through channel k:

$$\theta_{ijk} = P(c_{ijk} = 1 | e_{ij} = 1) = \frac{p_{ik}p_{jk}}{1 - \sum_{k=1}^{K}(1 - p_{ik}p_{jk})}.$$

- Size of channel: $S_k = \sum_{i=1}^{N_n} p_{ik}$

- Expected number of connections through channel k:

$$C_{ik} = \mathbb{E}\left[\sum_{j \neq i} c_{ijk} | G\right] = \sum_{j \neq i} e_{ij}\theta_{ijk}$$

Fitting Model

- If latent edges from nodes to channels known, would have closed form solution for channel frequency parameters:

$$\hat{p}_{ik} = \sum_{j \neq i}^{N_n} \tilde{e}_{ikj} / (N_n - 1)$$

- Expected value of latent edges, given current parameter values:

$$P(\tilde{e}_{ikj} = 1 | e_{ij} = 1) = p_{ik} p_{jk} + p_{ik}(1 - p_{jk}) \left(1 - \prod_{k' \neq k} (1 - p_{ik'} p_{jk'}) \right)$$

$$P(\tilde{e}_{ikj} = 1 | e_{ij} = 0) = p_{ik} - p_{ik} p_{jk}$$

- We have necessary ingredients for an EM algorithm!

Algorithm

Algorithm 1: Simple ECM Algorithm

Result: Fixed point estimate of $N \times K$ matrix p
Adjacency Matrix e; K;
N = nrow(e);
p = RandomUniform(min = 0, max = 1, nrow = N, ncol = K);
maxIters = 1,000; iter = 0; tol = 10^{-4}; maxDiff = tol + 1;
while *iter < maxIters & tol > maxDiff* **do**
  pOld = p;
  iter++;
  for *i in 1:N* **do**
    for *k in 1:K* **do**
      for *j in 1:N* **do**

$$\tilde{e}_{ijk} = \begin{cases} 0 & \text{if } i = j \\ P(\tilde{e}_{ikj} | e[i,j] = 1) & \text{else if } e[i,j] = 1 \\ P(\tilde{e}_{ikj} | e[i,j] = 0) & \text{otherwise} \end{cases}$$

      end
      p[i,k] = $\frac{\sum_{j=1}^{N} \tilde{e}_{ikj}}{N-1}$;
    end
  end
  maxDiff = max(| p - pOld |) ;
end
return (p)

Directly computed, $O(K^2E + K(N^2 - E))$

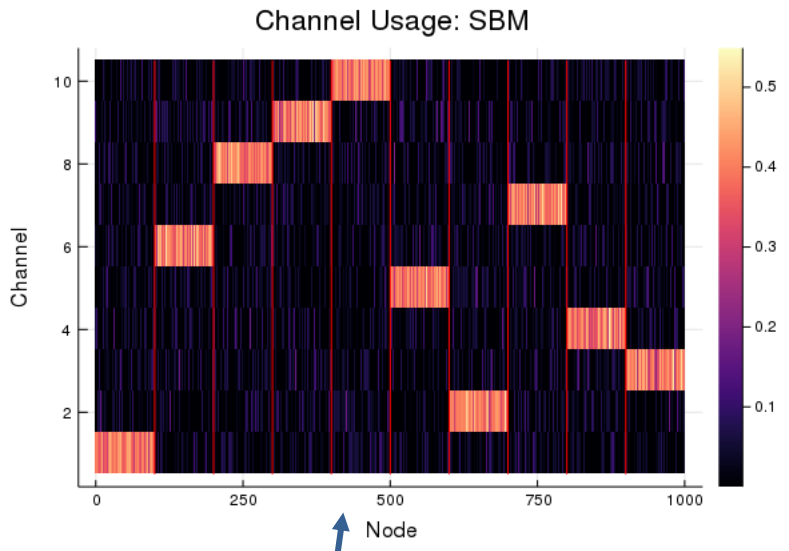
Caching + active set algorithm reduces to $O(K(E + N))$

Algorithm 2: Cached ECM Algorithm

Result: Fixed point estimate of $N \times K$ matrix p
Edge list E s.t. $E[i][j] \equiv j^{th}$ index of node sharing j^{th} edge with node i;
ReverseMapping R, such that $E[j][R[i][j]] = E[i][j]$;
p = RandomUniform(min = 0, max = 1, nrow = N, ncol = K);
pBar = ColumnMean(p);
initialize edge probability list P, where $P[i][j] = P(e_{iE[i][j]} = 1)$;
maxIters = 1,000; iter = 0;
tol = 10^{-4}; pTol = $10^{-}10$; maxDiff = tol + 1;
while *iter < maxIters & tol > maxDiff* **do**
  pOld = p;
  iter++;
  for *i in 1:N* **do**
    # Extract node indices and edge probabilities for nodes attached to node i
    edges = E[i];
    edgePs = P[i];
    for *k in 1:K* **do**
      pik = p[i,k];
      if *pik < pTol* **then**
        skip;
      end
      # Compute contributions from nodes with and without edges to node i
      edgeContribution = 0.0;
      noEdgeContribution = N * pik * (1 - pBar[k]) - pik *(1 - pik);
      for *j in 1:length(edges)* **do**
        pjk = p[j,k];
        noEdgeContribution -= pik * (1 - pjk);
        edgeContribution += pik * (pjk + (1 - pjk) * (1 - $\frac{1 - \text{edgePs[j]}}{1 - \text{pik} * \text{pjk}}$));
      end
      # Compute update
      pikNew = (edgeContribution + noEdgeContribution) / (N - 1);
      p[i,k] = pikNew;
      # Update column averages of p
      pikDiff = pikNew - pik;
      pBar[k] += pikDiff / N;
      for *j in 1:length(edges)* **do**
        # Update edge probabilities, including transpose
        pjk = p[j,k];
        newEdgeP = 1 - $\frac{(1 - \text{edgePs[j]}) * (1 - \text{pikNew} * \text{pjk})}{1 - \text{pik} * \text{pjk}}$;
        P[i][j] = newEdgeP;
        P[E[i][j]][R[i][j]] = newEdgeP;
      end
    end
    maxDiff = max(| p - pOld |);
  end
end
return p;

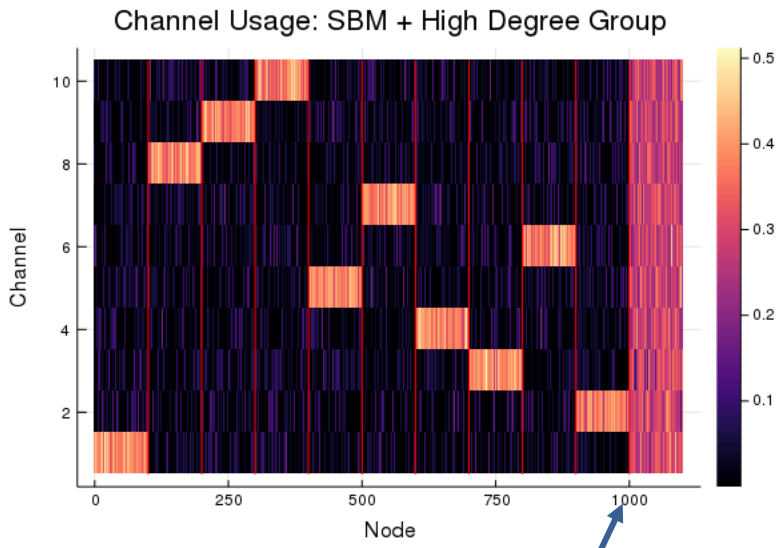
Example: Stochastic Block Models

10 Blocks
100 Nodes per Block
$p_{in} = 0.25, p_{out} = 0.025$



Recovers SBM form; each channel captures one block

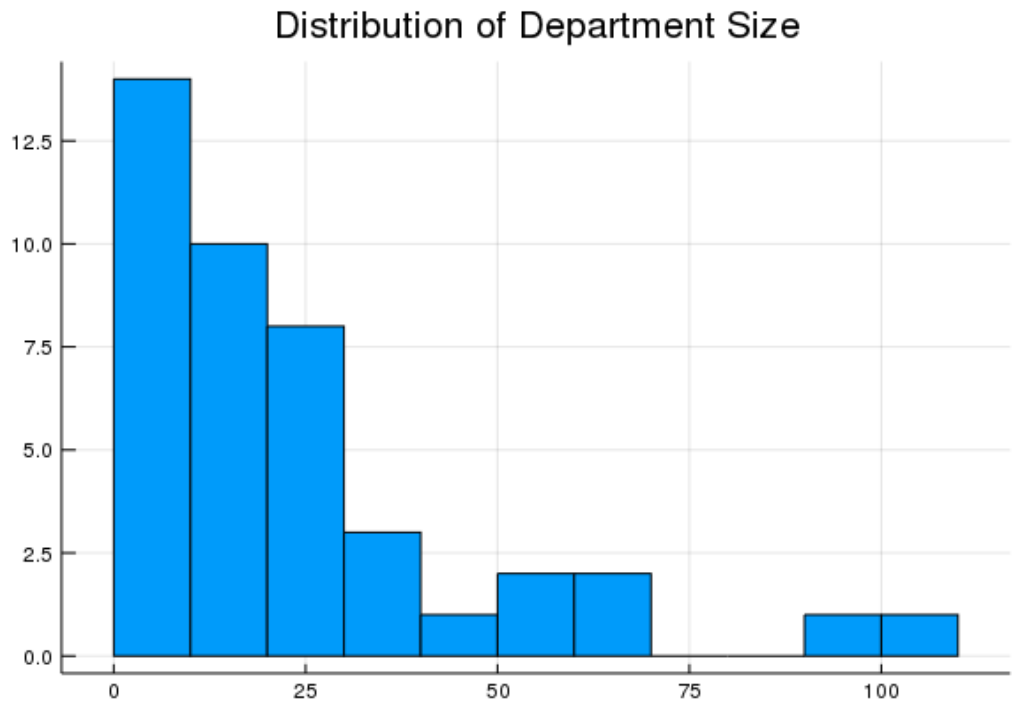
Augmented with new block with $p_{in} = 0.25, p_{out} = 0.25$



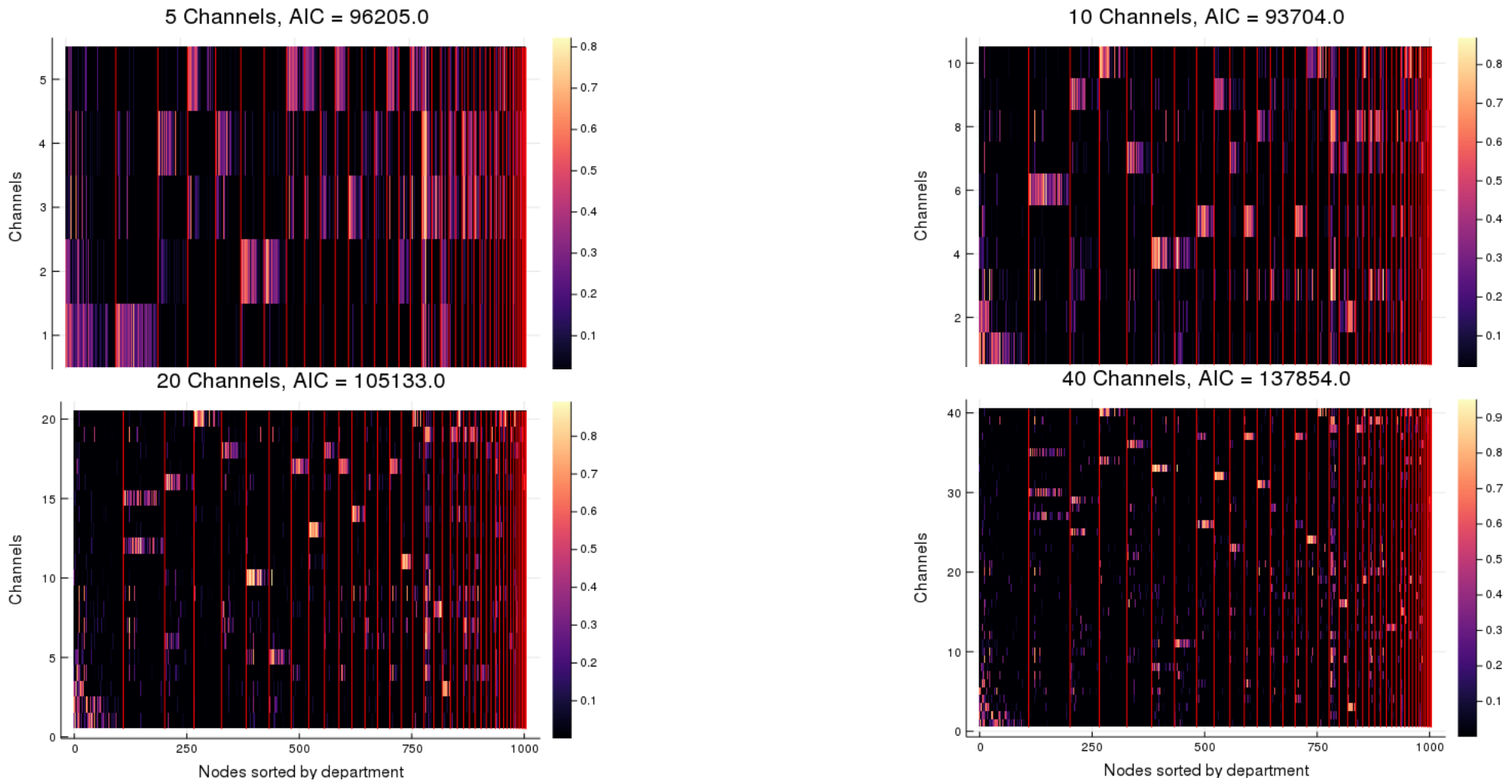
High degree nodes strongly attached to *all* channels

Example: Eu-Core Email Network

- Network built from emails within a university
- Faculty are nodes
- Emails sent are edges
- 1005 Nodes
- 24,929 Edges
- Labels: 42 departments, ranging from 109 to 1 members

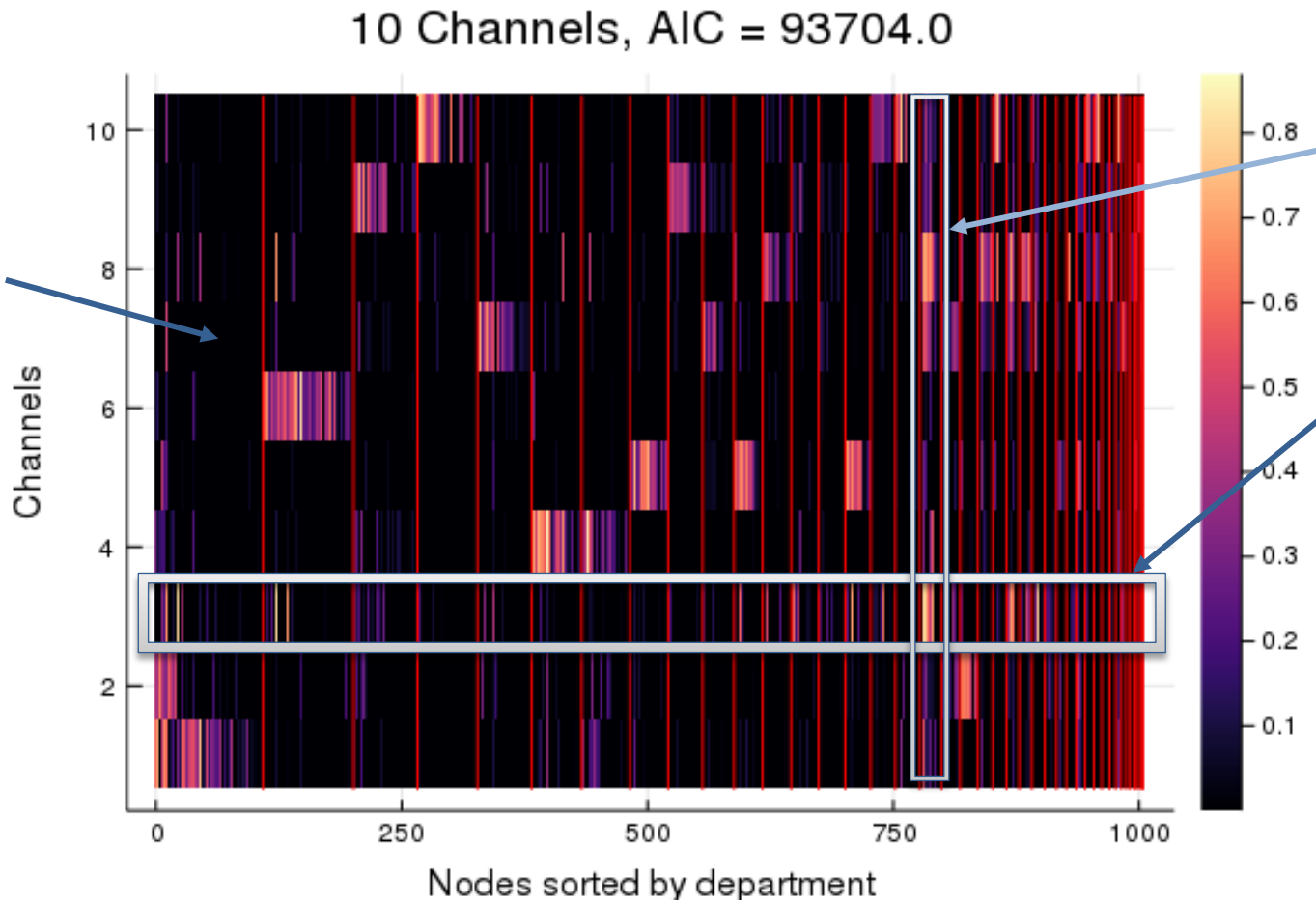


Example: Eu-Core Email Network



Example: Eu-Core Email Network

Many departments fairly well described by SBM-like structure



One department has strong attachments to *many* channels.

One channel especially strongly attached to this department, most departments have very small number (often 1) people attached to this channel.

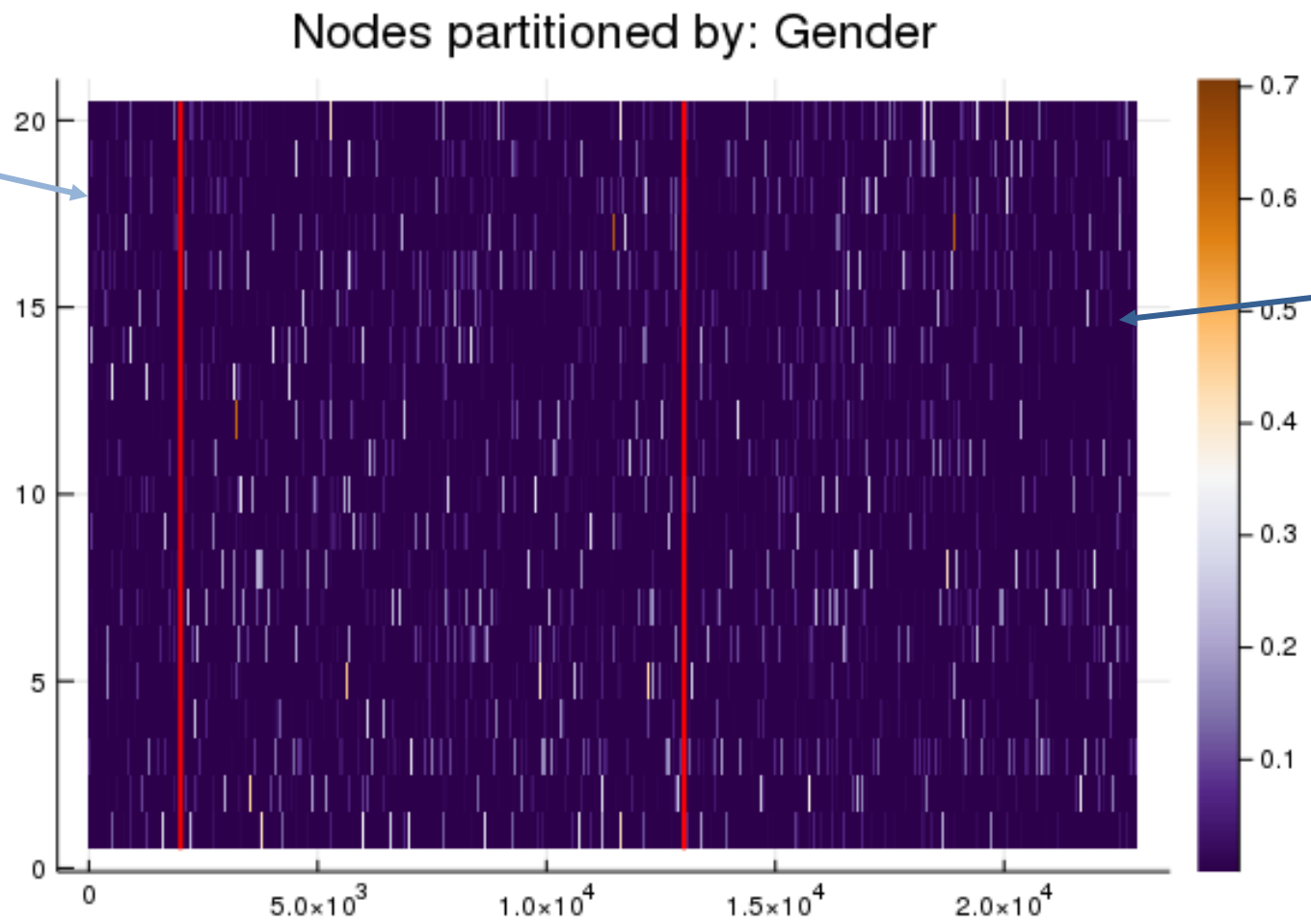
Perhaps administrative group?

Example: Facebook 100 UC Berkeley

- Snapshot of Facebook data from 2005
- Subsample of UC Berkeley only students
  - Number of nodes = 22,937
  - Number of edges = 852,445
- Metadata
  - Gender
  - Student/Faculty
  - Major
  - Year

Example: Facebook 100 UC Berkeley

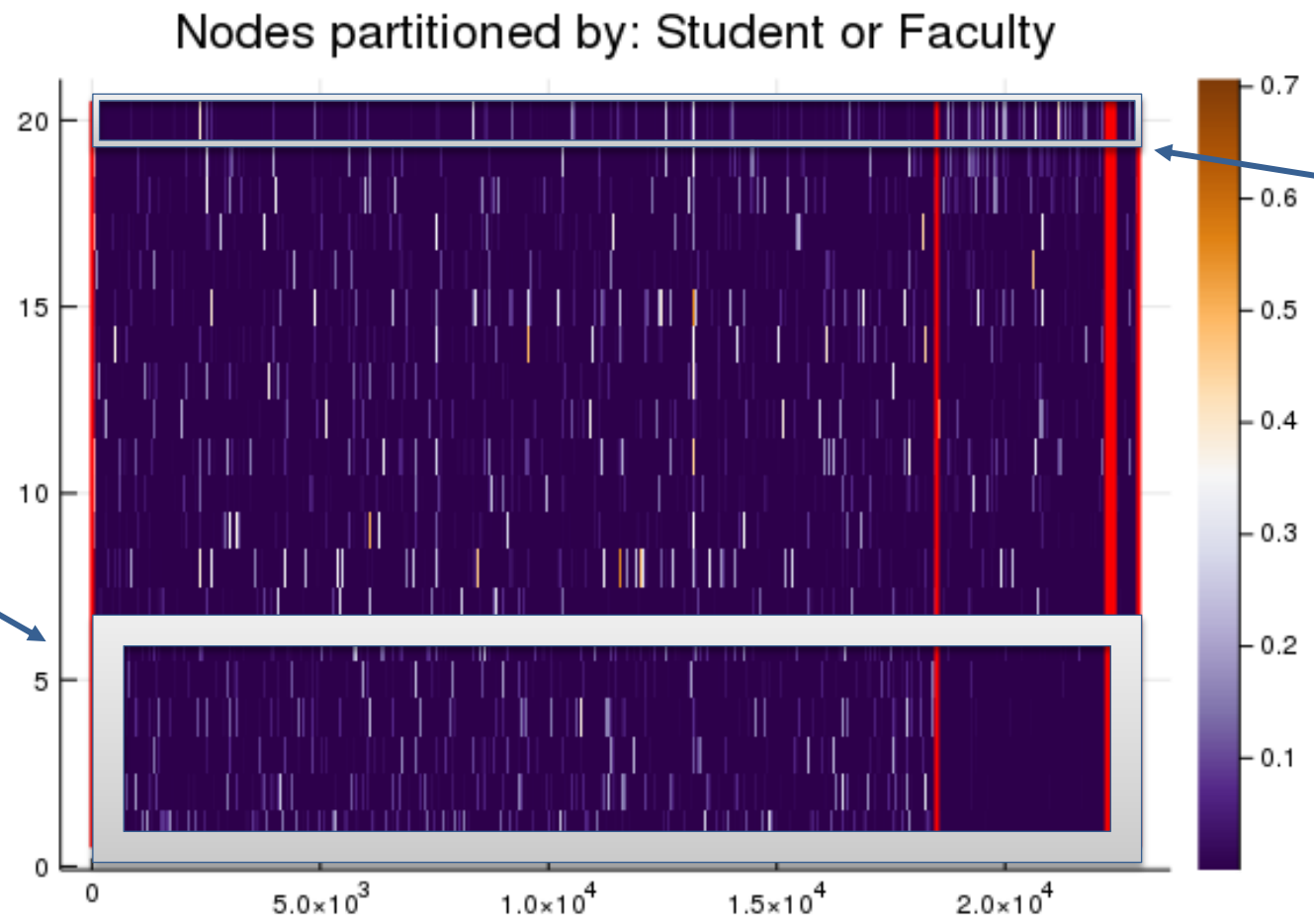
Metadata missing



Channel frequency use appears roughly independent of gender for all channels

Mathematically, should not expect latent channels to pick up on heterophilic tendencies of a network

Example: Facebook 100 UC Berkeley



Channel heavily used by faculty, lightly used by students

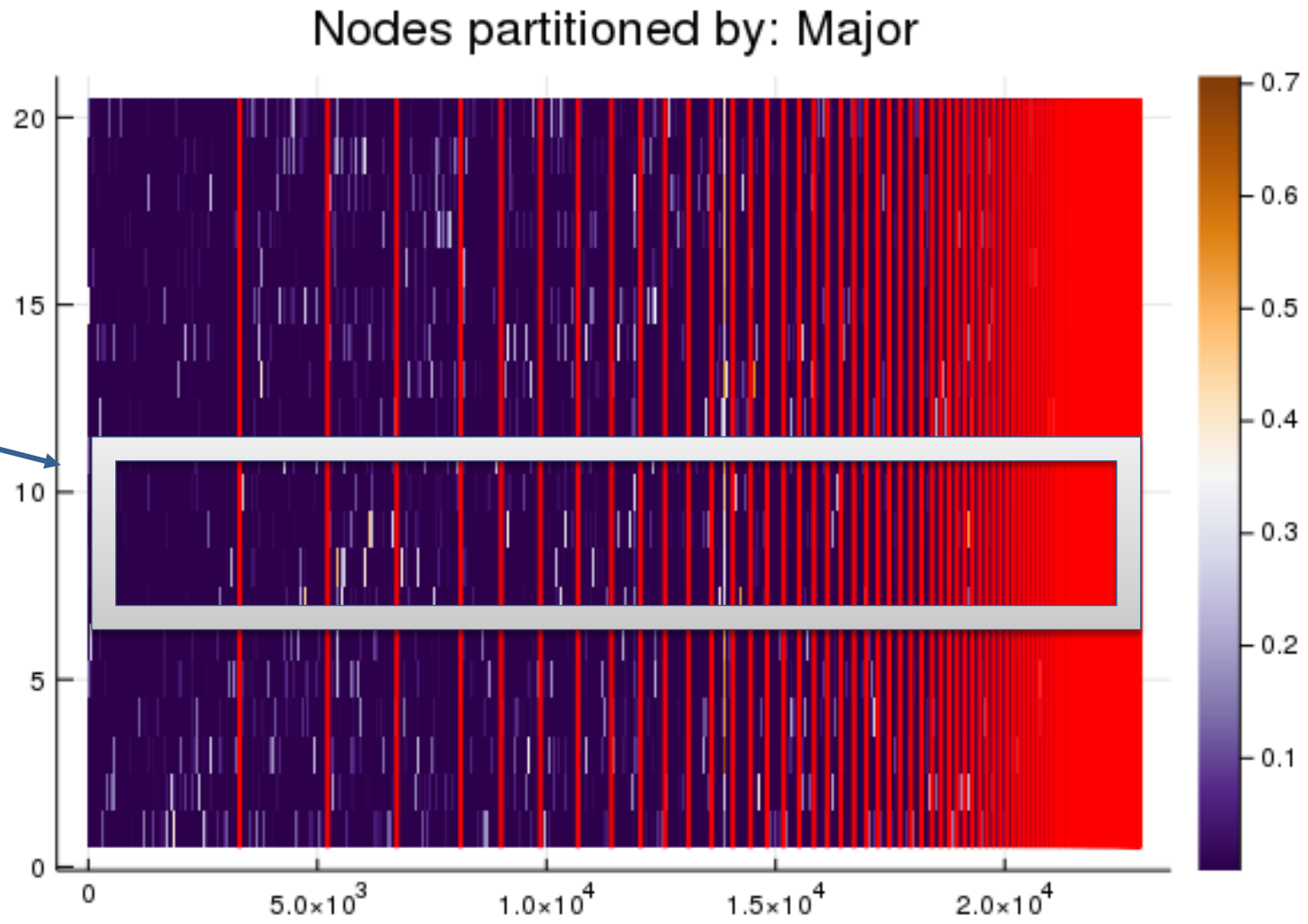
Channels exclusively used by students

Example: Facebook 100 UC Berkeley

No channels appear *exclusive* to any single major

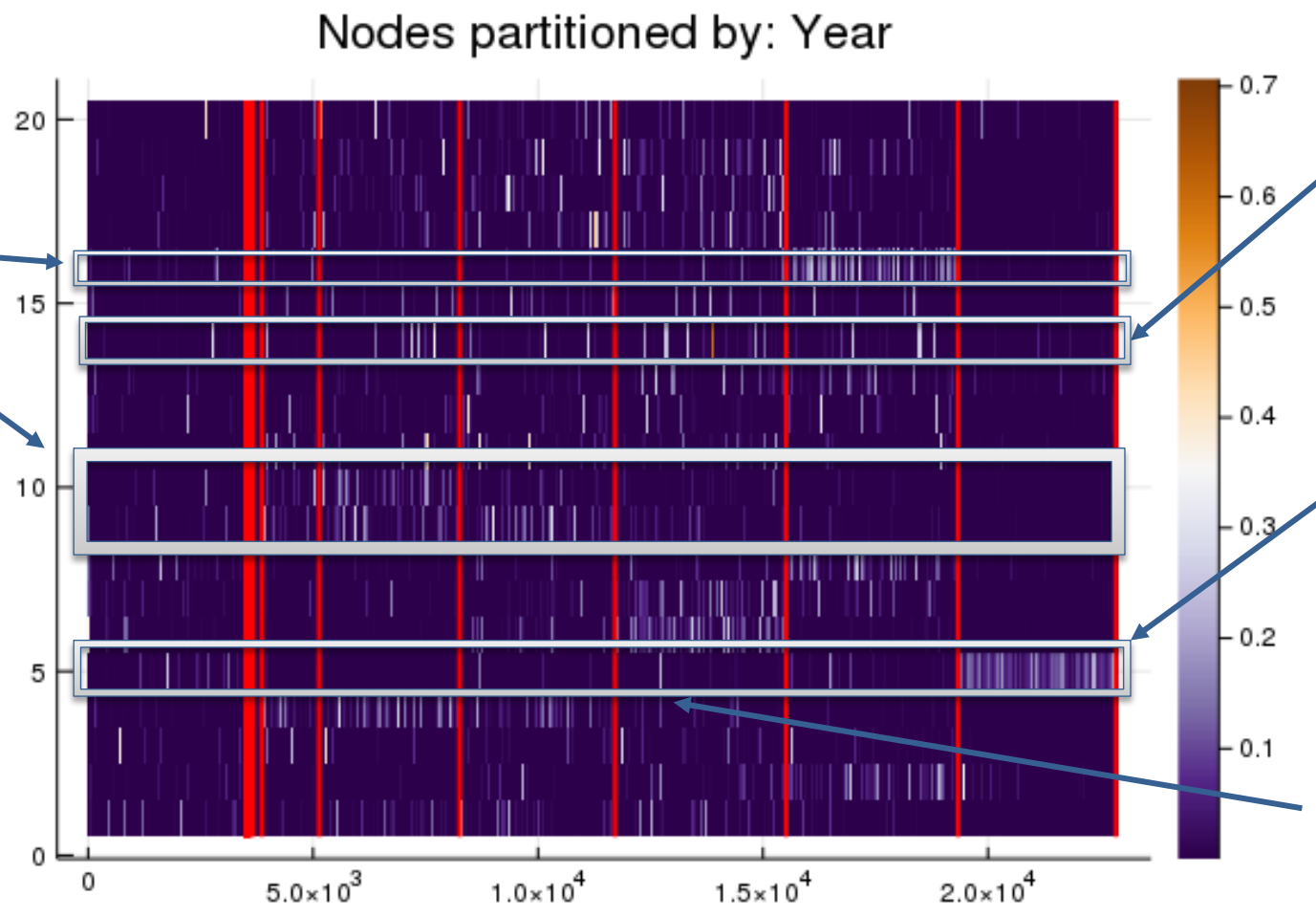
Many channels have no attachment to various majors

Hypothesis: channels not capturing major precisely, but perhaps major hierarchy: i.e., {math, physics, ...}, {sociology, history, ...}



Example: Facebook 100 UC Berkeley

Several channels *strongly* associated to year



Channels with strong attachments nearly independent of year. Why?

Freshman almost exclusively attached to single channel

Very few upperclassmen also attached to freshman channel, but also strongly attached to other hubs

Future Work

- **Applying to interesting applications**
- Accelerating EM-algorithm
  - Iterations computationally cheap
  - *Lots* (1000s) of iterations required
    - Specialized EM-variants may be able to reduce required iterations
- Expanding to further graph types
    - Weighted graphs
    - Directed graphs

Example: Eu-Core Email Network

- Can use hub-strength parameters to predict meta-data
- Plugged into glmnet to predict top 6 departments
- Approximately 88% class accuracy in validation set