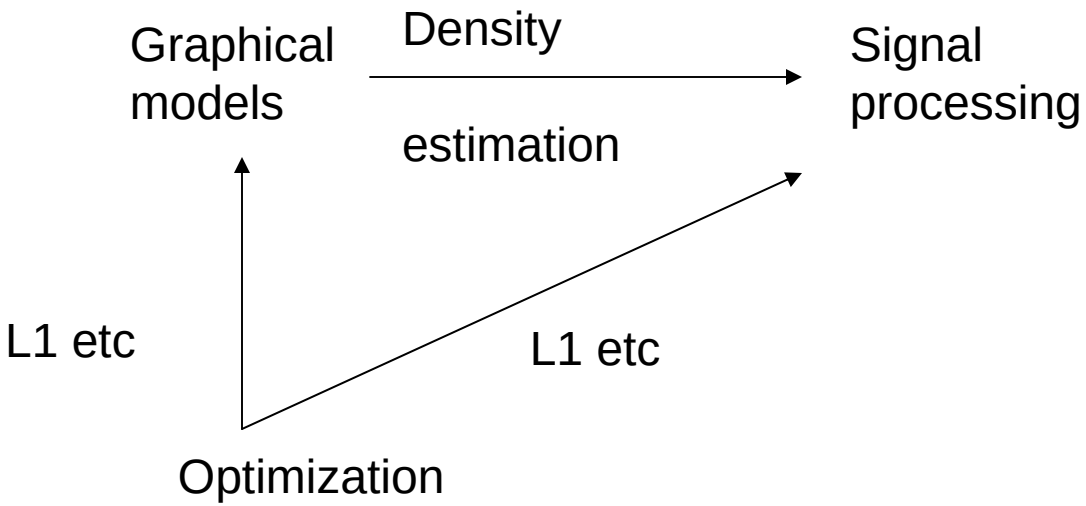
Learning Gaussian Graphical Models with Unknown Group Sparsity

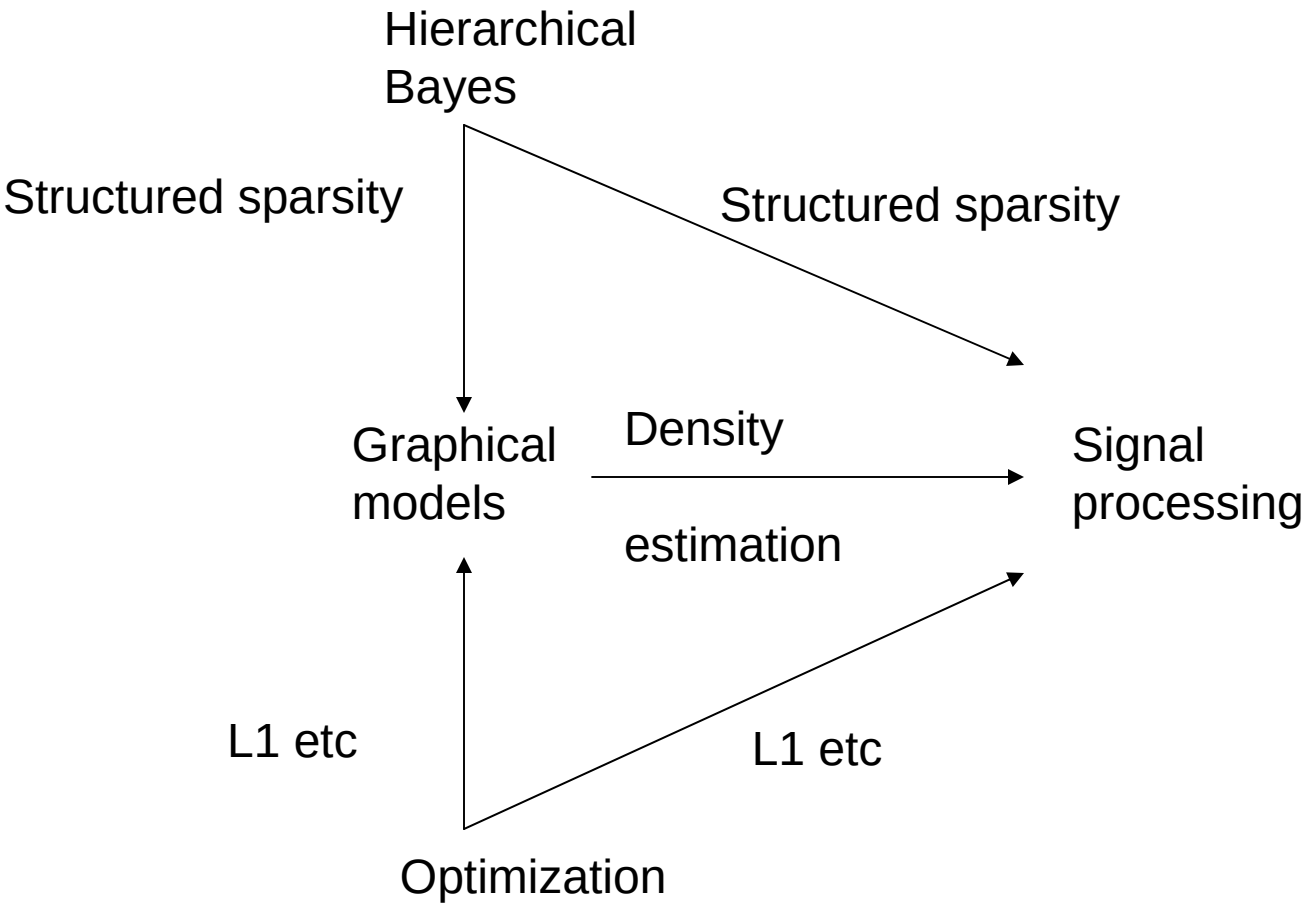
Kevin Murphy
Ben Marlin

Depts. of Statistics & Computer Science
Univ. British Columbia
Canada

Connections

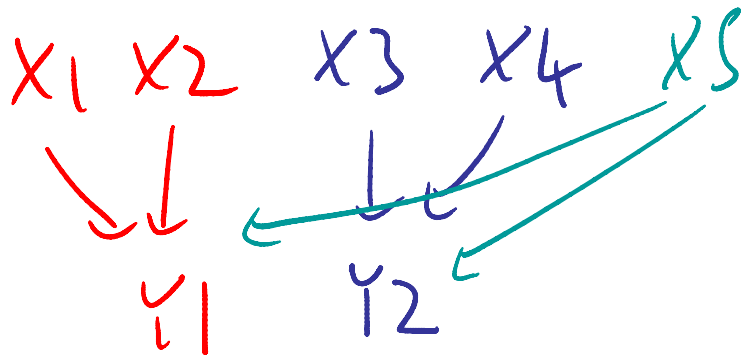


Connections

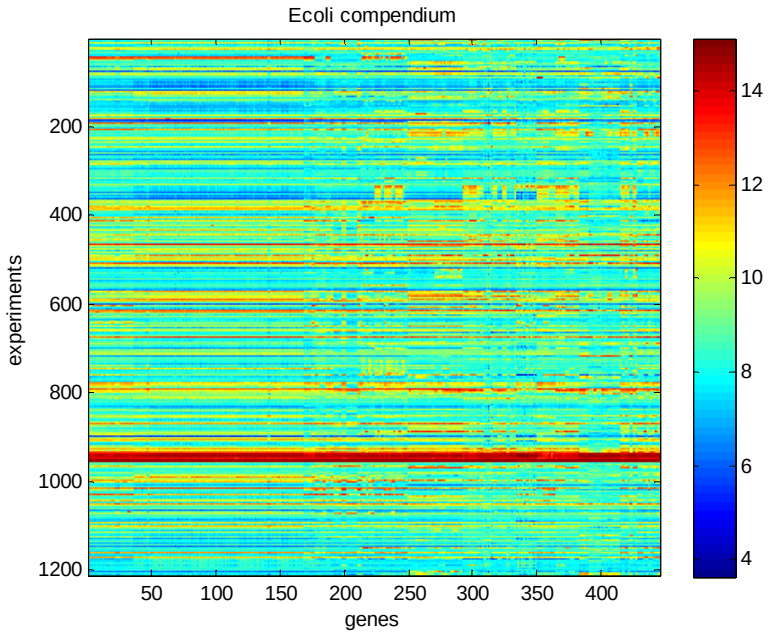


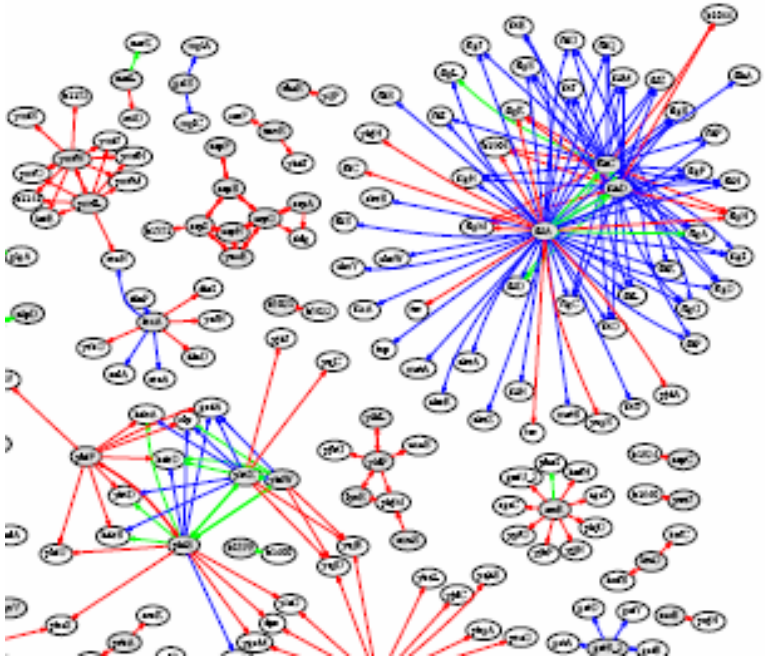
Structured sparsity

- Sparse models are often more interpretable and yield better prediction accuracy than dense ones
- Sometimes there is structure in the sparsity pattern itself
- Eg. Multi-task feature selection



Graphical model structure learning





n=1211, d=445

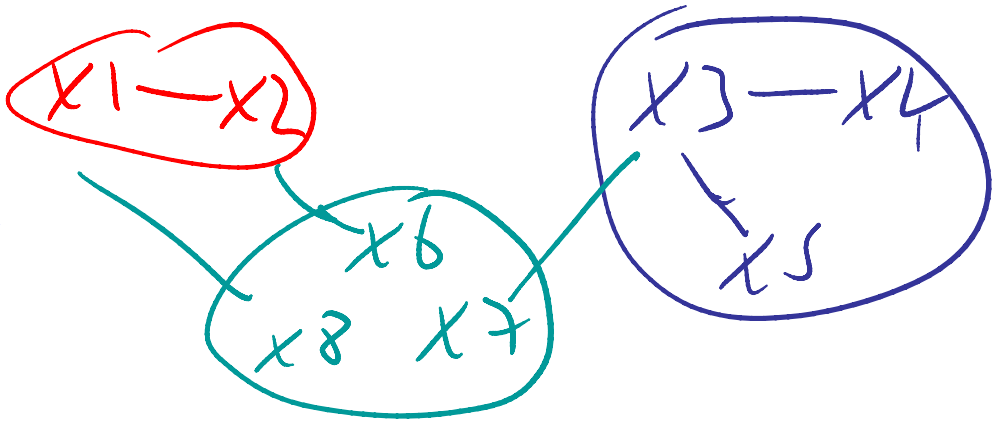
Goals:

- Learn interpretable structure
- Build better density models for high dimensional data

Structured sparsity in GM learning

- Similar types of nodes within a graph may share similar neighbors (stochastic blocks model)

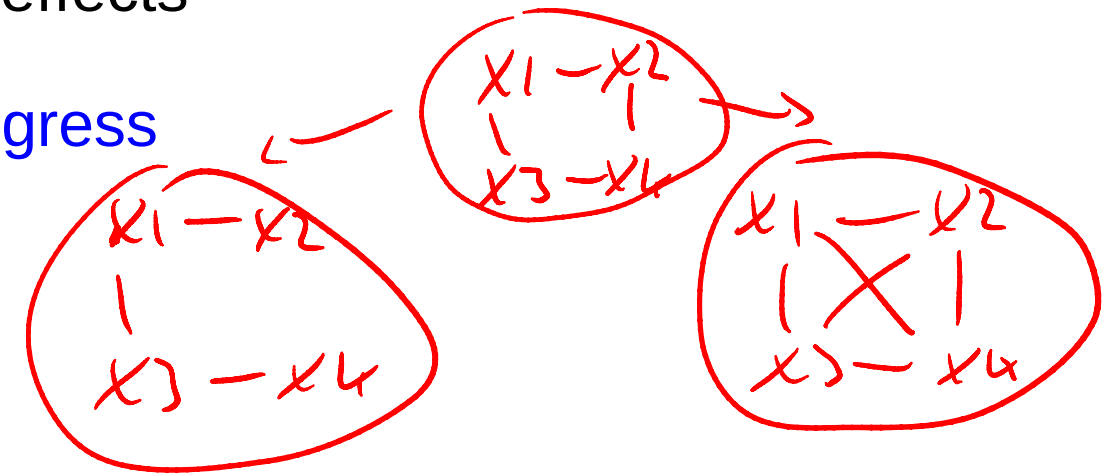
This talk



	G1	G2	G3
G1	0.0	0	0.5
G2	0.0	0.9	0.2
G3	0.5	0.2	0.0

- Multi-level analysis: individual graph = population graph plus random effects

Work in progress

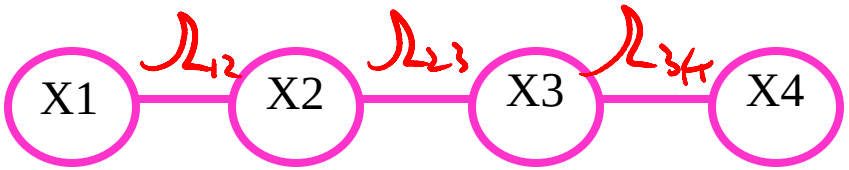


Gaussian graphical models (GGMs)

- Graph encodes structural zeros in precision matrix (conditional independence relationships)

$$p(\mathbf{x}|G, \boldsymbol{\mu}, \boldsymbol{\Omega}^{-1}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Omega}^{-1}), \quad \Omega_{jk} = 0 \iff G_{jk} = 0$$

$$\boldsymbol{\Omega} = \begin{pmatrix} \Omega_{11} & \Omega_{12} & 0 & 0 \\ \Omega_{21} & \Omega_{22} & \Omega_{23} & 0 \\ 0 & \Omega_{32} & \Omega_{33} & \Omega_{34} \\ 0 & 0 & \Omega_{43} & \Omega_{44} \end{pmatrix}$$



$$X_j \perp X_{-j} | X_{N_j}$$

Markov property

Dempster, Speed, Lauritzen, et al.

Maximum likelihood estimation

- To find the MLE given *known* G, maximize the following convex objective

$$
\begin{aligned}
p(\mathcal{D}|\mathbf{\Omega}) &\propto |\mathbf{\Omega}|^{n/2} \exp(-\tfrac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i - \overline{\mathbf{x}})^T \mathbf{\Omega}(\mathbf{x}_i - \overline{\mathbf{x}})) \\
&= |\mathbf{\Omega}|^{n/2} \exp(-\frac{n}{2}\mathrm{tr}(\mathbf{S\Omega})) \\
\ell(\mathbf{\Omega}) &\stackrel{\text{def}}{=} \frac{2}{n}\log p(\mathcal{D}|\mathbf{\Omega}) = \log\det\mathbf{\Omega} - \mathrm{tr}(\mathbf{S\Omega})
\end{aligned}
$$

- Subject to

$$\mathbf{\Omega} \succ \mathbf{0}, \Omega_{jk} = 0 \iff G_{jk} = 0$$

Outline

- Previous work: L1 and group L1 approaches to learning structure
- Our contribution: unknown groups

Sparse Cholesky Decomposition

- Given a known ordering, we can learn a sparse DAG, with regression weights W

$$X_j = \mu_j + \sum_{k \in \pi_j} w_{jk}(X_k - \mu_k) + \sqrt{v_j} Z_j$$

$$\mathbf{\Sigma}^{-1} = (\mathbf{I} - \mathbf{W})^T \mathbf{D}^{-1} (\mathbf{I} - \mathbf{W}) \stackrel{\text{def}}{=} \mathbf{T}^T \mathbf{D}^{-1} \mathbf{T}$$

$$\mathbf{T} = \begin{pmatrix} 1 & & & & \\ -w_{21} & 1 & & & \\ -w_{32} & -w_{31} & 1 & & \\ \vdots & & & \ddots & \\ -w_{d1} & -w_{d2} & \dots & -w_{d,d-1} & 1 \end{pmatrix}$$

Sparse dependency networks

- We can regress each node on all the others, and impose an L1 regularizer (Meinshausen & Buhlmann)
- ie we optimize the penalized pseudo likelihood

$$\prod_{n=1}^{N} p(x_{nj}|x_{n,-j}, w_j, \sigma_j^2) + \lambda \sum_{j=1}^{D} ||w_{j,:}||_1$$

Graphical lasso

- Put L1 penalty on precision matrix

$$\max_{\mathbf{\Omega} \succ \mathbf{0}} \log\det(\Omega) - \mathrm{tr}(S\Omega) - \sum_{i=1}^{D}\sum_{j=1}^{D} \lambda_{ij}|\Omega_{ij}|$$

- Semi-definite program. $\lambda_{jj} \geq 0, \lambda_{jk}^{max} = |\hat{\Sigma}_{jk}|$
- Banerjee proposed a coordinate descent method, where we optimize one row/column at a time by solving a QP, O(T d^4) time, T=#iter.
- Friedman et al. improved this by showing that each subproblem is a weighted lasso problem. Still O(T d^3).

Banerjee et al, Friedman et al, Lee et al.

Group graphical lasso

- Sometimes nodes can be grouped, eg genes in the same pathway.
- We would like to learn a graph which is block sparse.
- We can use the group L1 method to get a new convex objective:

$$\max_{\mathbf{\Omega} \succ \mathbf{0}} \log\det(\Omega) - \text{tr}(S\Omega) - \sum_k \sum_l \lambda_{k,l} ||\{\mathbf{\Omega}_{z_k, z_l}\}||_p$$

- Common p-norms:

$$||\mathbf{v}||_\infty = \max_i |v_i|$$

$$||\mathbf{v}||_2 = \sqrt{\sum_i v_i^2}$$

Duchi et al, UAI 2008

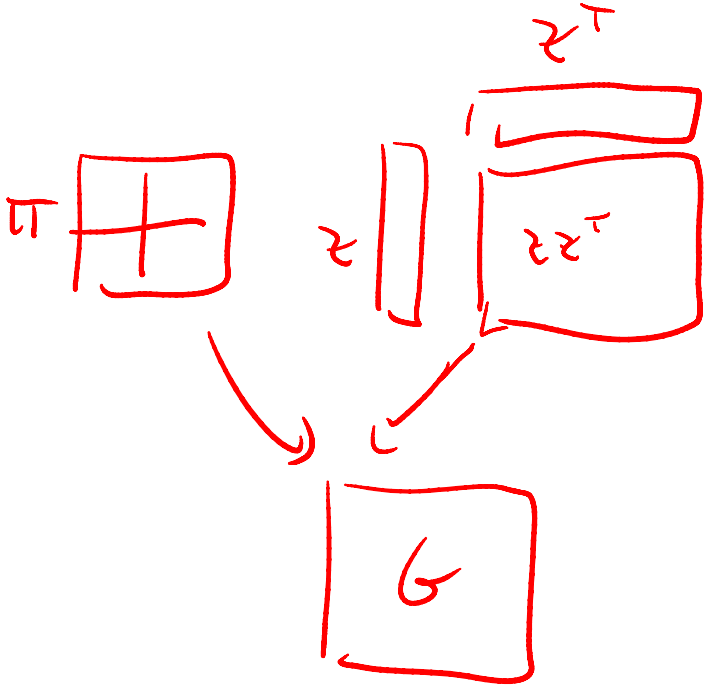
Outline

- Previous work: L1 and group L1 approaches
- Our contribution: unknown groups

Stochastic blocks model (SBM)

- We associate nodes with *latent* types or groups, $z_i \in \{1,\ldots,K\}$
- The probability of an edge from node i to node j depends on their types (say red and blue), and on the probability of a red – blue edge.
-

$$
\begin{aligned}
G_{i,j} &\sim \text{Ber}(\pi_{z_i,z_j}) \\
z_i &\sim \text{Mu}(\theta, 1) \\
\pi_{k,k'} &\sim \text{Beta}(a_{k,k'}, b_{k,k'}) \\
\theta &\sim \text{Dir}(\frac{\alpha}{K})
\end{aligned}
$$



Nowicki & Snijders

Using SBM to define the prior on Ω

- The SBM generates a block sparse undirected graph G via assignments z
- We would like G to induce a block sparse penalty matrix on Ω.
- But the normalization "constant" is intractable to compute (not a problem for fixed z).
- Graphs not just independent regressions!

$$p(\Omega|\lambda, z) = \frac{1}{Z} pd(\Omega) \prod_k \prod_l \exp(-\lambda_{k,l} ||\mathbf{\Omega}_{z_k, z_l}||_p)$$

$$Z(\lambda, z) = \int_{pd} \prod_k \prod_l \exp(-\lambda_{k,l} ||\mathbf{\Omega}_{z_k, z_l}||_p) d\Omega$$

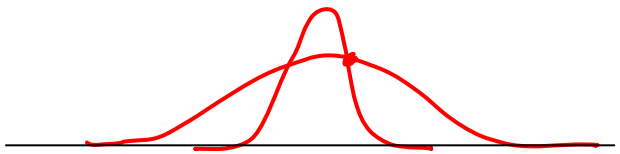
2 approaches to avoid intractable Z

- Use dependency networks
- Lower bound the objective

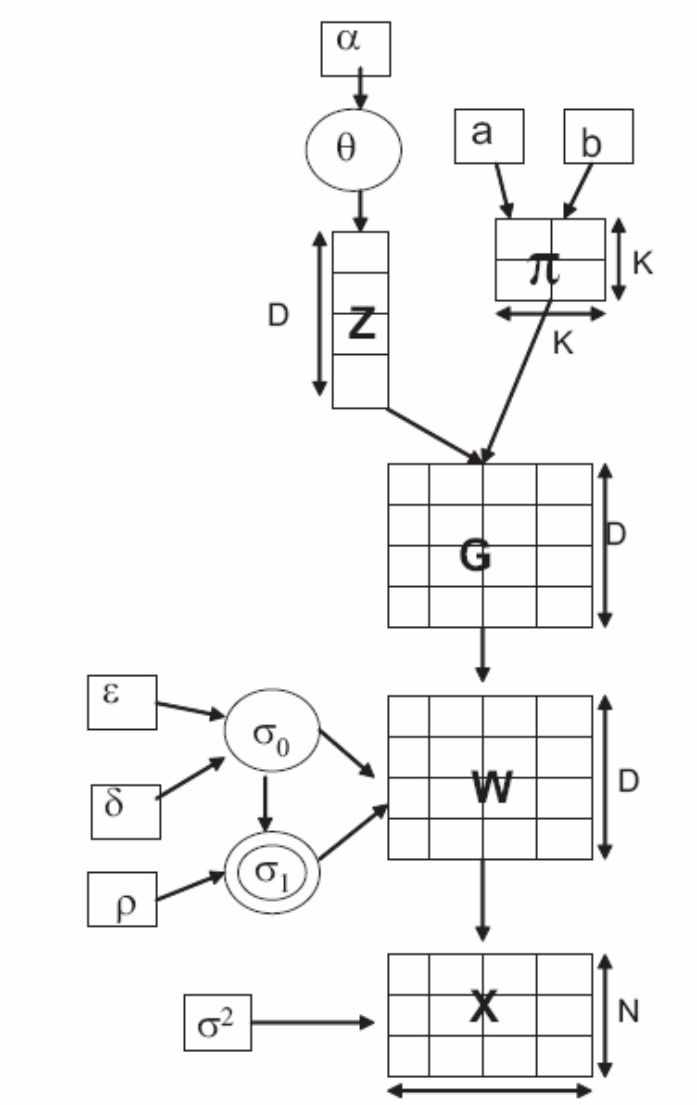
Block sparse dependency networks

- Dependency networks are a very fast way to learn graph structures.
- They do not define a valid joint density, but can be useful for data visualization.
- It is simple to impose block sparsity on them: use a SBM to define G, and use this indicator matrix to define a spike and slab prior

$$
\begin{aligned}
G &\sim SBM \\
w_{i,j} &\sim \mathcal{N}(0, \sigma_0^2)^{G_{i,j}} \mathcal{N}(0, \sigma_1^2)^{1-G_{i,j}} \\
x_{i,n} &\sim \mathcal{N}(w_i^T x_{-i,n}, \sigma^2)
\end{aligned}
$$



Full model



Inference

- We use variational mean field, with a point estimate of W (which is $O(D^2)$ in size)

$$
\begin{aligned}
q(\pi_{k,k'}) &= \text{Beta}\left(\sum_{d=1}^{D}\sum_{d'=1}^{D}\phi_{d,k}\phi_{d',k'}\gamma_{d,d'} + a_{k,k'},\ \sum_{d=1}^{D}\sum_{d'=1}^{D}\phi_{d,k}\phi_{d',k'}(1-\gamma_{d,d'}) + b_{k,k'}\right)\\
q(\theta) &= \text{Dir}(\alpha_k + \sum_{d=1}^{D}\sum_{k=1}^{K} E[z_d = k])\\
q(1/\sigma_0^2) &= \text{Ga}\left(\epsilon + \frac{D(D-1)}{2},\ \delta + \sum_{d=1}^{D}\sum_{d'\neq d}\frac{w_{d,d'}^2}{2}\left(\frac{\gamma_{d,d'}}{\rho} + (1-\gamma_{d,d'})\right)\right)\\
\Lambda_d &= \text{diag}\left(\frac{\epsilon^*}{\delta^*}\left(\frac{\gamma_{d,d'}}{\rho} + (1-\gamma_{d,d'})\right)\right)\\
w_d &= (\sigma^2\Lambda_d + \sum_n X_{-dn}^T X_{-dn})^{-1}(\sum_n X_{dn}X_{-dn})\\
\overline{\pi}_{k,k} &= E[\log \pi_{k,k'}] = \Psi(a_{k,k'}^*) - \Psi(a_{k,k'}^* + b_{k,k'}^*)\\
\hat{\pi}_{k,k} &= E[\log(1-\pi_{k,k'})] = \Psi(b_{k,k'}^*) - \Psi(a_{k,k'}^* + b_{k,k'}^*)\\
q(G_{d,d'}) &= \text{Ber}\left(G_{d,d'}|\text{logistic}\left[\sum_{k=1}^{K}\sum_{k'=1}^{K}\phi_{d,k}\phi_{d',k'}(\overline{\pi}_{k,k'} - \hat{\pi}_{k,k'}) + \tfrac{1}{2}(\kappa_0-\kappa_1)(w_{d,d'}^2 + w_{d',d}^2) - \tfrac{1}{2}\log\rho\right]\right)\\
q(z_d) &= \text{Multi}\left(z_d|\text{softmax}(\sum_{d'\neq d}\sum_{k'=1}^{K}\phi_{d',k'}(\gamma_{d,d'}\overline{\pi}_{k,k'} + (1-\gamma_{d,d'})\hat{\pi_{k,k'}}) + \Psi(\alpha_k^*) - \Psi(\alpha^*))\right)\\
\text{softmax}_k(v) &= \frac{\exp(v_k)}{\sum_{k'=1}^{K}\exp(v_{k'})}
\end{aligned}
$$

Output of inference

- Clustering $p(z_i = k) = \phi_{ik}$
- Graph $p(G_{i,j'} = 1) = \gamma_{i,j}$
- Abstract graph $p(\pi_{k,l}) = \text{Beta}(a^*_{k,l}, b^*_{k,l})$
- Nuisance variables $\hat{W}, p(\sigma), p(\theta)$

Computational speed

- At each iteration, we solve D related ridge regression problems, each of size D, costs $O(D^4)$.

$$\Lambda_d = \mathrm{diag}\left(\frac{\epsilon^*}{\delta^*}\left(\frac{\gamma_{d,d'}}{\rho} + (1 - \gamma_{d,d'})\right)\right)$$

$$w_d = (\sigma^2 \Lambda_d + \sum_n X_{-dn}^T X_{-dn})^{-1} (\sum_n X_{dn} X_{-dn})$$

- No obvious way to use standard matrix tricks to improve this (invert diagonal + up/downdate)
- So we do adaptive scheduling: only update top 10% of nodes, sorted by $|\Lambda_d|$
- In Matlab: for N=174,D=667, was 50 sec per iter., now is 5 sec

Model selection

- We use spectral clustering to propose splits
- The graph for cluster k is defined by

$$
\begin{aligned}
S &= \{i : z_i = k\} \\
H &= |W(S,S)| + 0.5|W(S,\overline{S})||W(S,\overline{S})^T|
\end{aligned}
$$

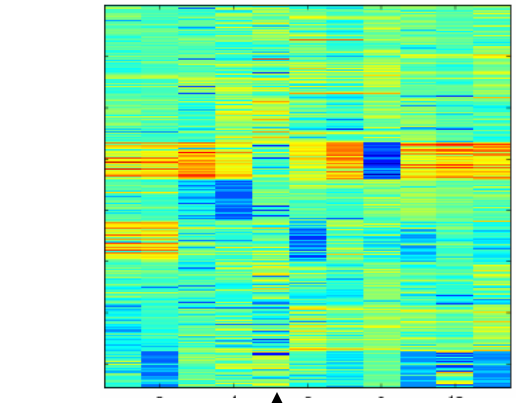
- We propose splitting each cluster and use the variational free energy as an objective to select the best

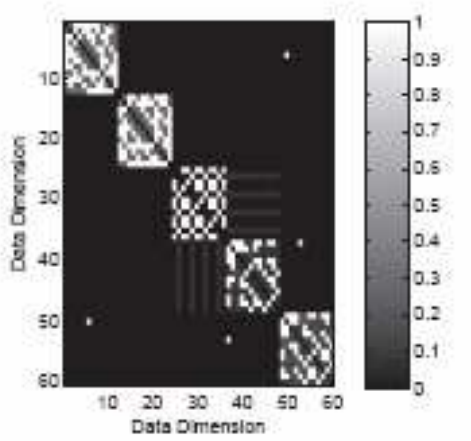
Estimating Ω

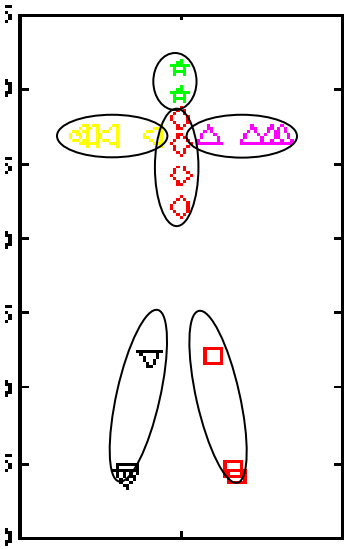
- Once we have estimated the depnet, we want to convert it to a GGM so we have a proper joint density model
- We use the inferred clusterings $\text{argmax}_k\ p(z_d=k)$ to define the groups, and use group L1 penalized MAP estimation to estimate Ω.
- We estimate λ by CV.

Results on mocap data

- D=60 (20 markers in 3d), N=100



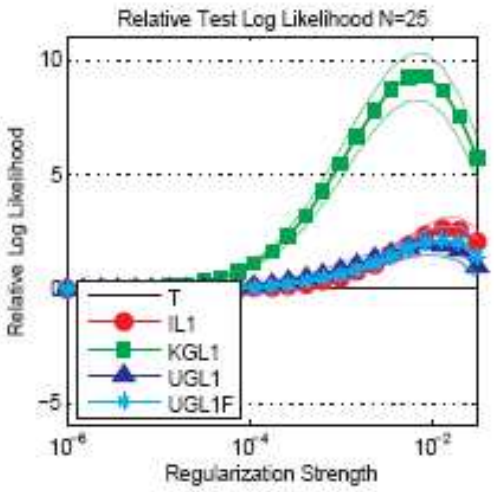




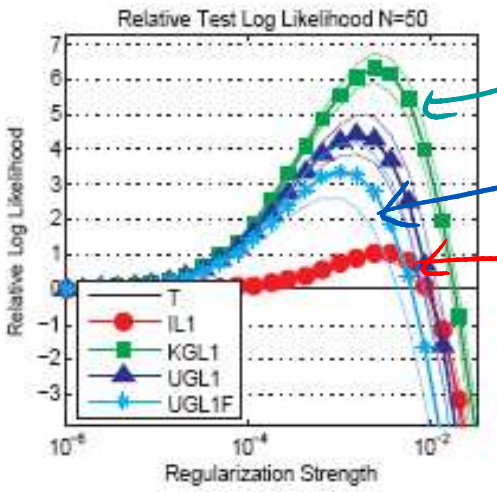
Test set likelihood vs λ for different N

Baseline

$$\hat{\Sigma} = S + \epsilon I$$



(a) N=25

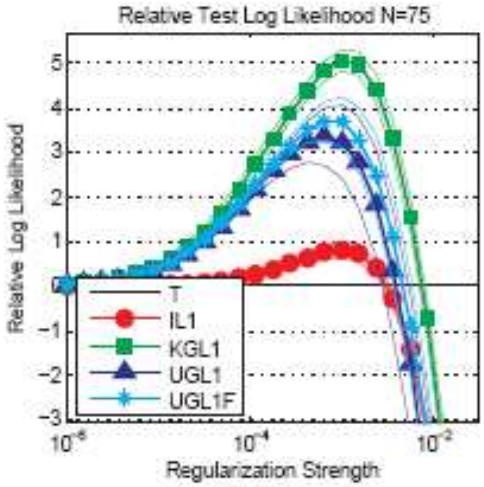


(b) N=50

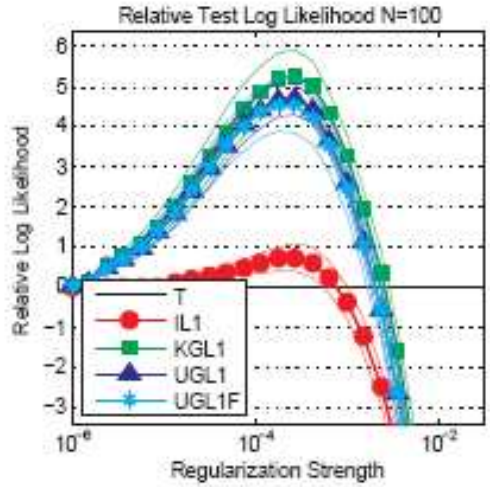
known groups

estimated

no groups



(c) N=75



(d) N=100

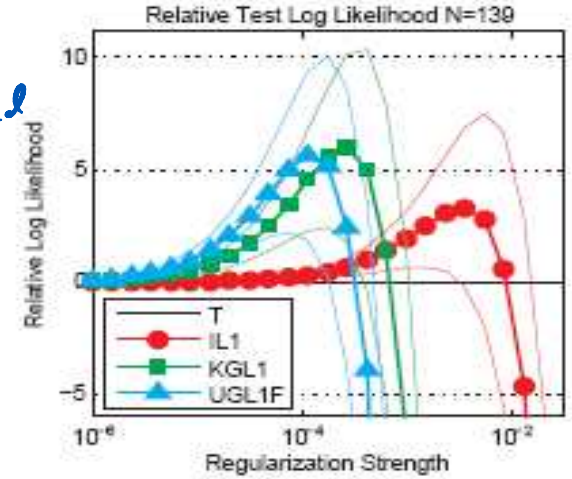
Results on gene expression data

- N=174,D=667

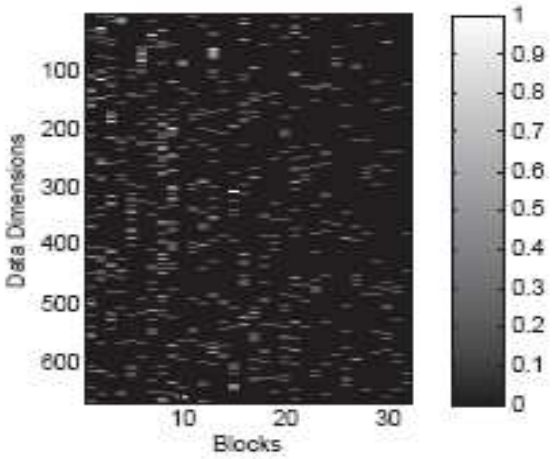
estimated

known

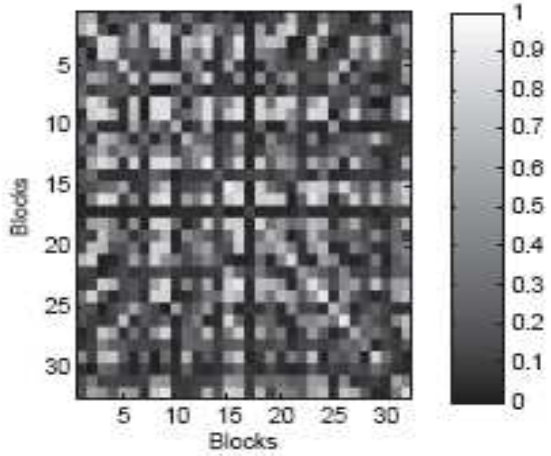
none



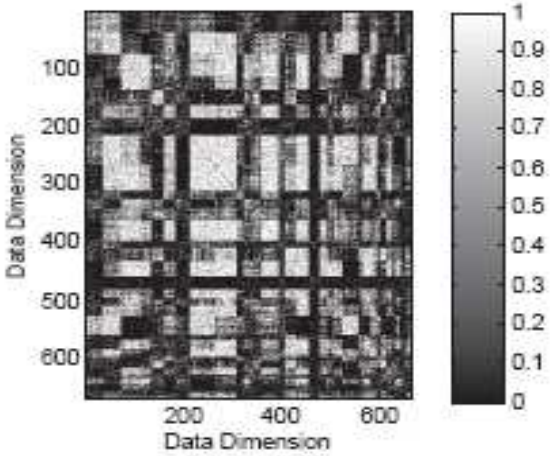
(a) Log likelihood vs λ



(b) Clustering $E(z)$



(c) Block connectivity $E(\pi)$



(d) Graph $E(G)$

2 approaches to avoid intractable Z

- Use dependency networks
- Lower bound the objective

Z (Independent L1)

$$P(\Sigma^{-1}) \propto \mathrm{pd}(\Sigma^{-1}) \prod_{i=1, j \geq i}^{D} \exp(-\lambda_{ij} |\Sigma_{ij}^{-1}|)$$

$$\mathcal{Z} = \int_{\mathbb{R}^{PD}} \prod_{i=1, j \geq i}^{D} \exp(-\lambda_{ij} |X_{ij}|) dX$$

$$\lambda_{ij} = \lambda_0 [z_i \neq z_j] + \lambda_1 [z_i = z_j]$$

Between within

Upper bound for Z

$$
\begin{aligned}
\mathcal{Z} &= \int_{\mathbb{R}^{PD}} \prod_{i=1}^{D} \prod_{j \geq i}^{D} \exp(-\lambda_{ij}|X_{ij}|) dX \\
&\leq \int_{\mathbb{R}^{SP}} \prod_{i=1}^{D} \prod_{j \geq i}^{D} \exp(-\lambda_{ij}|X_{ij}|) dX \\
&= \prod_{i=1}^{D} \int_{0}^{\infty} \exp(-\lambda_{ii}|x|) dx \\
&\quad \cdot \prod_{i=1}^{D} \prod_{j>i}^{D} \int_{-\infty}^{\infty} \exp(-\lambda_{ij}|x|) dx \\
&= \prod_{i=1}^{D} \frac{1}{\lambda_{ii}} \cdot \prod_{i=1}^{D} \prod_{j>i}^{D} \frac{2}{\lambda_{ij}}
\end{aligned}
$$

- We integrate over symmetric matrices with positive elements on the diagonal.

EM algorithm

E-Step:

$$\phi_{ik} \leftarrow \frac{\theta_k \prod_{j \neq i} \left(\lambda_0 \exp(-\lambda_0 |\Sigma_{ij}^{-1}|)\right)^{1-\phi_{jk}} \left(\lambda_1 \exp(-\lambda_1 |\Sigma_{ij}^{-1}|)\right)^{\phi_{jk}}}{\sum_{k=1}^{K} \theta_k \prod_{j \neq i} \left(\lambda_0 \exp(-\lambda_0 |\Sigma_{ij}^{-1}|)\right)^{1-\phi_{jk}} \left(\lambda_1 \exp(-\lambda_1 |\Sigma_{ij}^{-1}|)\right)^{\phi_{jk}}}$$

M-Step:

$$\theta_k \leftarrow \frac{\sum_{i=1}^{D} \phi_{ik}}{\sum_{k=1}^{K} \sum_{i=1}^{D} \phi_{ik}}$$

$$\lambda_{ij} \leftarrow \lambda_0 (1 - \sum_{k=1}^{K} \phi_{ik}\phi_{jk}) + \lambda_1 (\sum_{k=1}^{K} \phi_{ik}\phi_{jk})$$

$$\Sigma^{-1} \leftarrow \arg\min_{S \succ 0} - \log |S| + \mathrm{tr}(\hat{\Sigma} S) + \sum_{i=1}^{D} \sum_{j \geq i}^{D} |S_{ij}|$$

Work in progress…

Conclusions

- Lots of recent interest in sparse GGMs using convex optimization
- Combining clustering and sparse graph estimation results in a non-convex problem (cf clustered multi-task learning)
- However, (variational) EM can be used, and can leverage the convex solvers in the M step
- Still looking for killer apps…