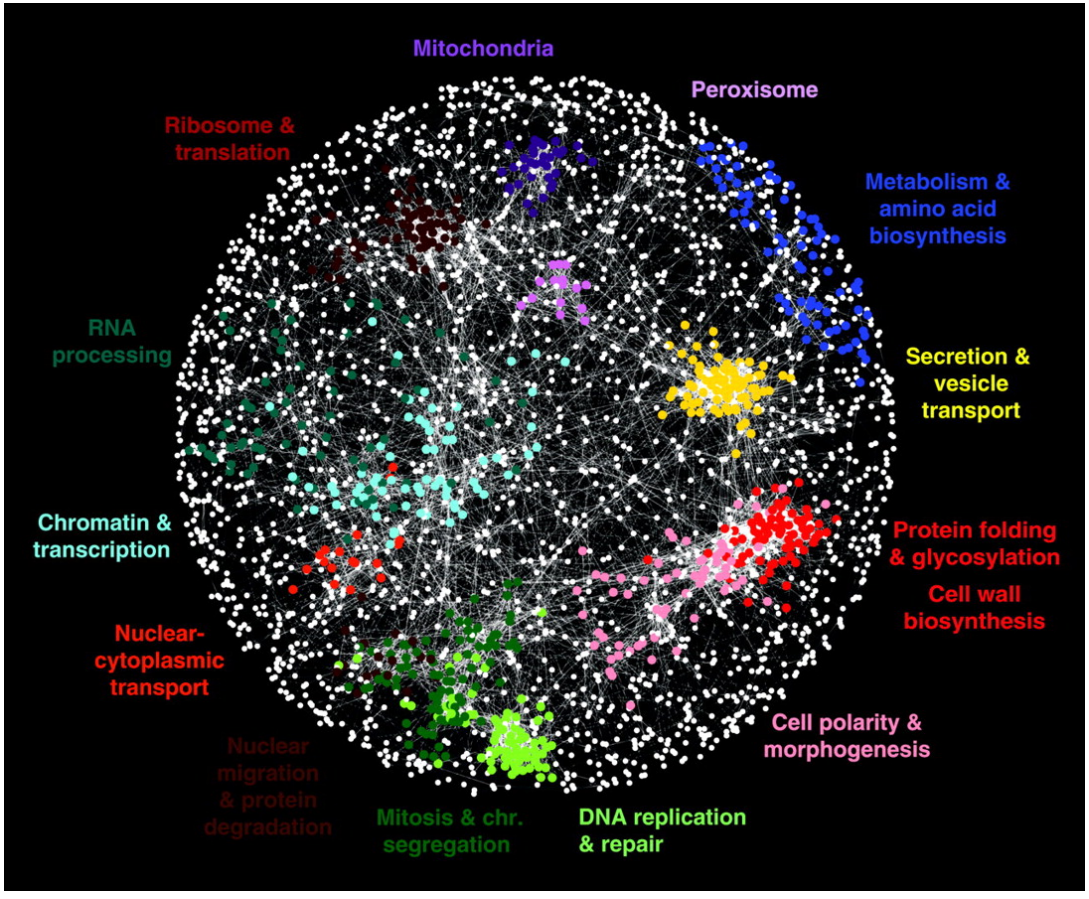
Multi-modal Structure Learning in High Dimensions for Integrative Genomics

Calvin McCarter

Joint work with Seyoung Kim

ML Lunch Seminar - October 5, 2015

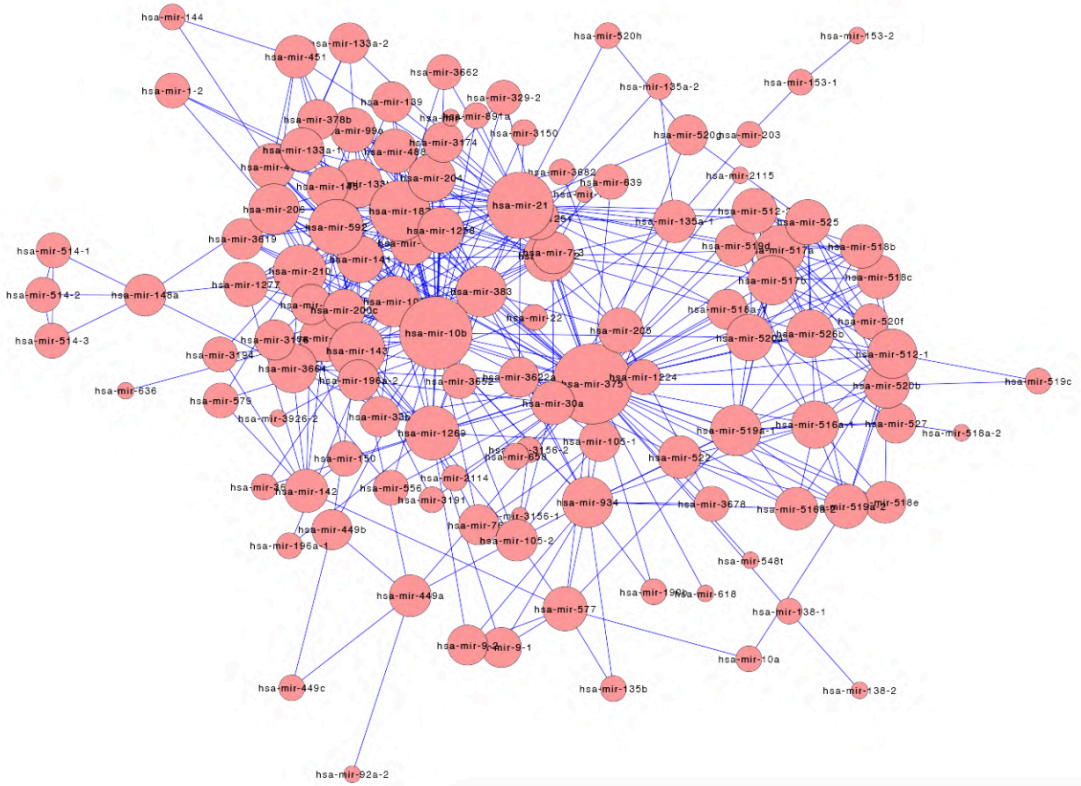
Network Inference for Scientific Discovery



Costanzo et al. "The genetic landscape of a cell." Science (2010).

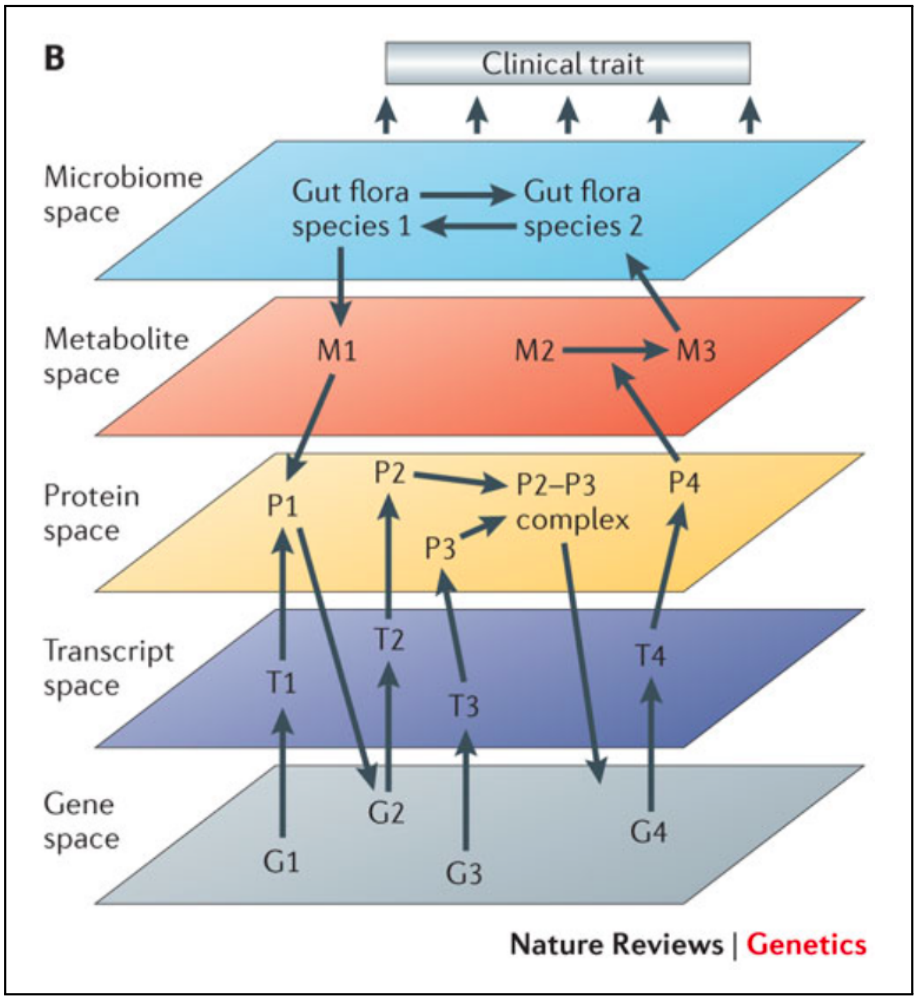
Background: Gaussian Graphical Model Estimation

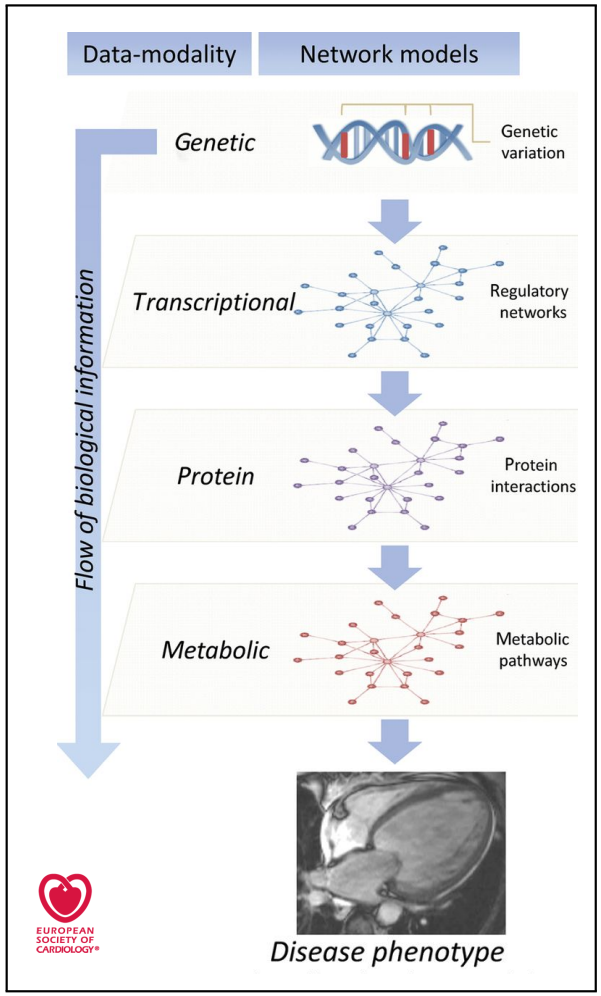
Graphical Lasso problem: $\min_{\Theta} -L(\Theta) + \|\Theta\|_1$



MicroRNA network learned from Cancer Genome Atlas data

Motivation: Multi-Modal Genomic Data

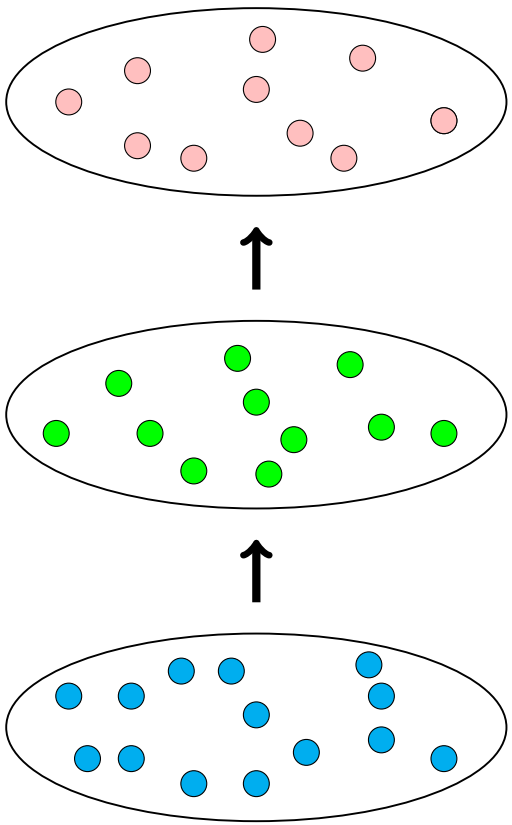




Structure Learning for Chain Graph Models

Input:

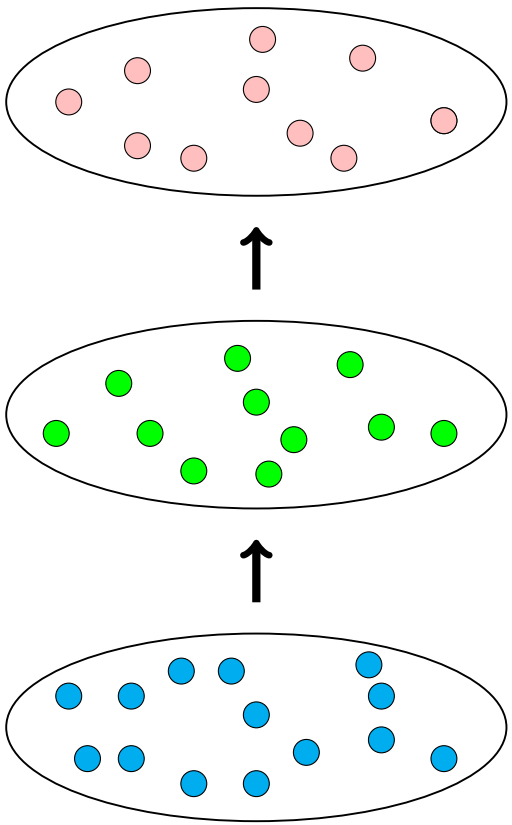
- A partition of variables into subsets
- A directed acyclic graph among subsets



Structure Learning for Chain Graph Models

Input:

- A partition of variables into subsets
- A directed acyclic graph among subsets



clinical variables

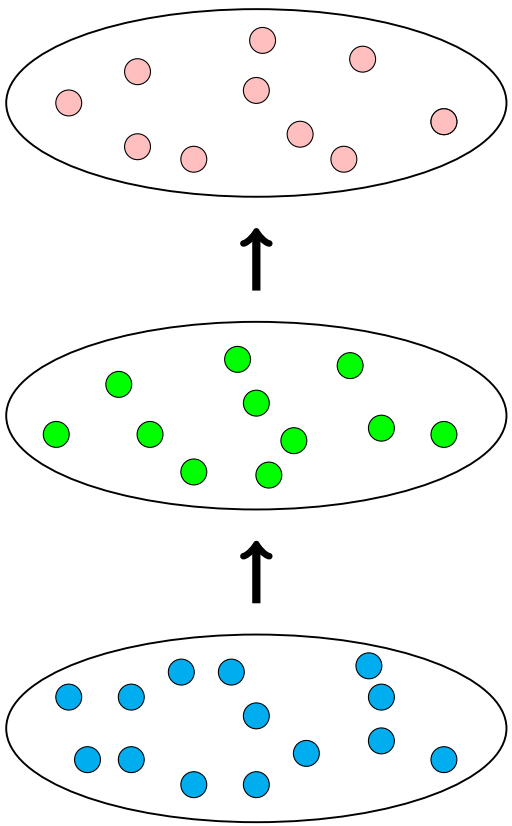
gene expression levels

mutations (SNPs)

Structure Learning for Chain Graph Models

Input:

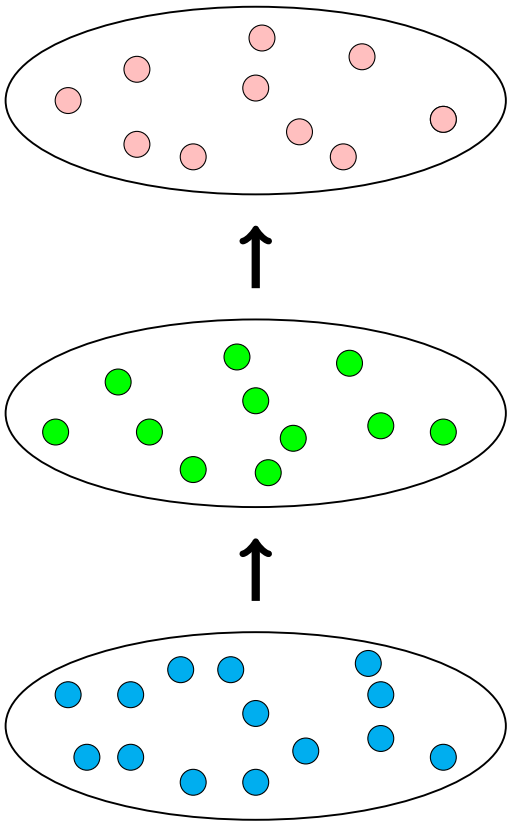
- A partition of variables into subsets
- A directed acyclic graph among subsets



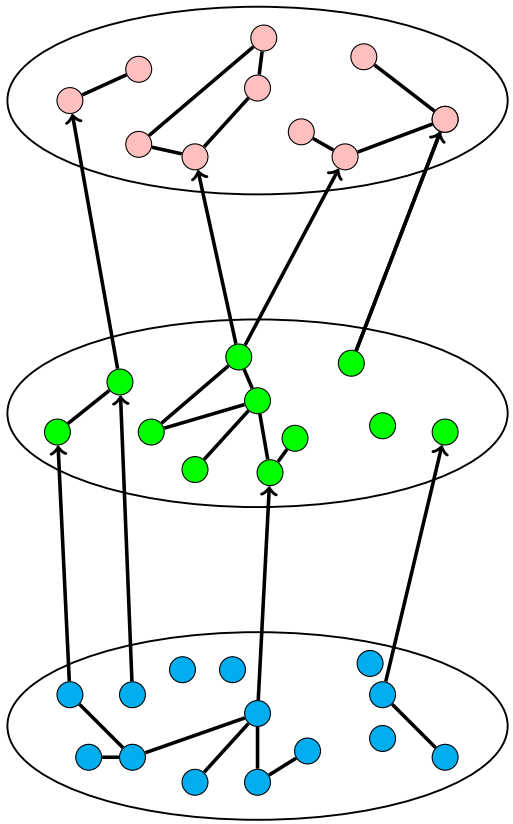
$$p(x, y, z) =$$
$$p(z|y)$$
$$\times$$
$$p(y|x)$$
$$\times$$
$$p(x)$$

Structure Learning for Chain Graph Models

Input:

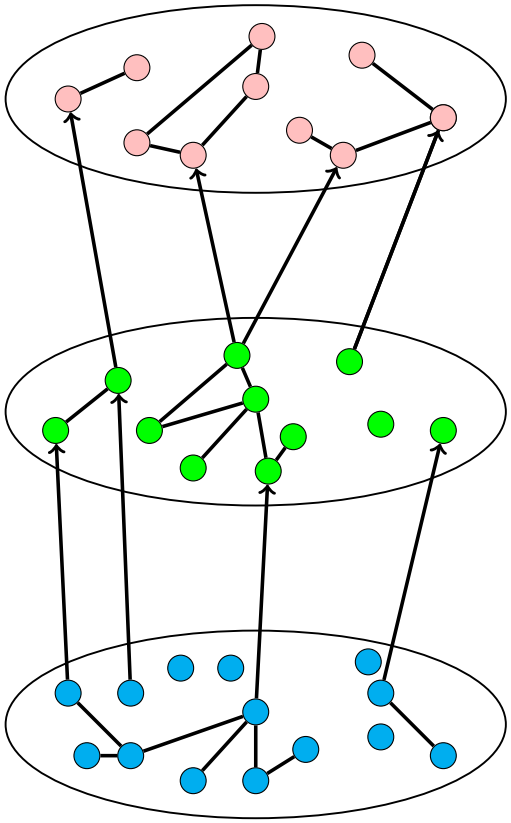


Learn from data:

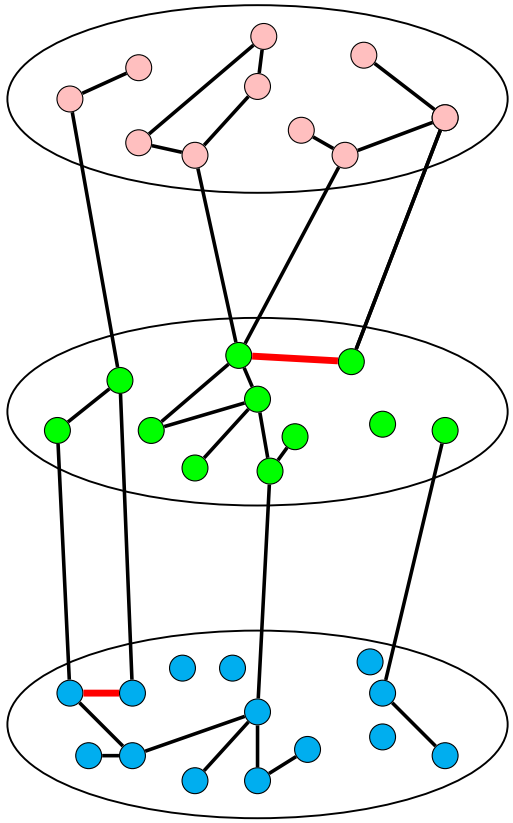


Markov Properties of Chain Graph Models

Original graph:



Moralized graph:



*Only works when component distributions specified as CRFs

Learning Gaussian Chain Graphs: Previous Approach

Model:

$$p(\mathbf{x}_\tau | \mathbf{x}_{\mathrm{pa}(\tau)}) = \mathcal{N}(\boldsymbol{B}_\tau \mathbf{x}_{\mathrm{pa}(\tau)}, \boldsymbol{\Lambda}_\tau^{-1})$$

Learning Gaussian Chain Graphs: Previous Approach

Model:

$$p(\mathbf{x}_\tau | \mathbf{x}_{\mathrm{pa}(\tau)}) = \mathcal{N}(\boldsymbol{B}_\tau \mathbf{x}_{\mathrm{pa}(\tau)}, \boldsymbol{\Lambda}_\tau^{-1})$$

- $\boldsymbol{B}$: directed edges, $\boldsymbol{\Lambda}$: undirected edges
- Standard Markov properties do not hold

Learning Gaussian Chain Graphs: Previous Approach

Model:

$$p(\mathbf{x}_\tau|\mathbf{x}_{\mathrm{pa}(\tau)}) = \mathcal{N}(\boldsymbol{B}_\tau \mathbf{x}_{\mathrm{pa}(\tau)}, \boldsymbol{\Lambda}_\tau^{-1})$$

Given n samples stored in $\mathbf{X}$:

$$\min \sum_\tau ((\mathbf{X}_\tau - \mathbf{X}_{\mathrm{pa}(\tau)}^T)\boldsymbol{\Lambda}_\tau(\mathbf{X}_\tau - \mathbf{X}_{\mathrm{pa}(\tau)}^T)^T) - n \log|\boldsymbol{\Lambda}_\tau|$$

$$+\lambda \sum_\tau \|\boldsymbol{B}_\tau\|_1 + \gamma \sum_\tau \|\boldsymbol{\Lambda}_\tau\|_1$$

Learning Gaussian Chain Graphs: Previous Approach

Given n samples stored in $\mathbf{X}$:

$$\min \sum_{\tau}((\mathbf{X}_{\tau} - \mathbf{X}_{\mathrm{pa}(\tau)}^{T})\mathbf{\Lambda}_{\tau}(\mathbf{X}_{\tau} - \mathbf{X}_{\mathrm{pa}(\tau)}^{T})^{T}) - n \log |\mathbf{\Lambda}_{\tau}|$$
$$+\lambda \sum_{\tau} \|\boldsymbol{B}_{\tau}\|_1 + \gamma \sum_{\tau} \|\mathbf{\Lambda}_{\tau}\|_1$$

- Bi-convex - multiple local optima
- Slow optimization algorithms

Learning Gaussian Chain Graphs: Our Approach

Conditional Gaussian Graphical Model (CGGM):

$$p(\mathbf{x}_\tau|\mathbf{x}_{\mathrm{pa}(\tau)}) = \exp\left(-\tfrac{1}{2}\mathbf{x}_\tau^T\mathbf{\Lambda}_\tau\mathbf{x}_\tau - \mathbf{x}_\tau^T\mathbf{\Theta}_{\tau,\mathrm{pa}(\tau)}\mathbf{x}_{\mathrm{pa}(\tau)}\right)/Z(\mathbf{x}_{\mathrm{pa}(\tau)})$$

Learning Gaussian Chain Graphs: Our Approach

Conditional Gaussian Graphical Model (CGGM):

$$p(\mathbf{x}_\tau|\mathbf{x}_{\mathrm{pa}(\tau)}) = \exp\left(-\tfrac{1}{2}\mathbf{x}_\tau^T\boldsymbol{\Lambda}_\tau\mathbf{x}_\tau - \mathbf{x}_\tau^T\boldsymbol{\Theta}_{\tau,\mathrm{pa}(\tau)}\mathbf{x}_{\mathrm{pa}(\tau)}\right)/Z(\mathbf{x}_{\mathrm{pa}(\tau)})$$

- $\boldsymbol{\Theta}$: directed edges, $\boldsymbol{\Lambda}$: undirected edges
- Standard Markov properties hold

Learning Gaussian Chain Graphs: Our Approach

Conditional Gaussian Graphical Model (CGGM):

$$p(\mathbf{x}_\tau|\mathbf{x}_{\mathrm{pa}(\tau)}) = \exp\left(-\tfrac{1}{2}\mathbf{x}_\tau^T\mathbf{\Lambda}_\tau\mathbf{x}_\tau - \mathbf{x}_\tau^T\mathbf{\Theta}_{\tau,\mathrm{pa}(\tau)}\mathbf{x}_{\mathrm{pa}(\tau)}\right)/Z(\mathbf{x}_{\mathrm{pa}(\tau)})$$

Given n samples stored in $\mathbf{X}$:

$$\min -L(\mathbf{X};\mathbf{\Theta},\mathbf{\Lambda}) + \lambda\sum_\tau \|\mathbf{\Theta}_{\tau,\mathrm{pa}(\tau)}\|_1 + \gamma\sum_\tau \|\mathbf{\Lambda}_\tau\|_1$$

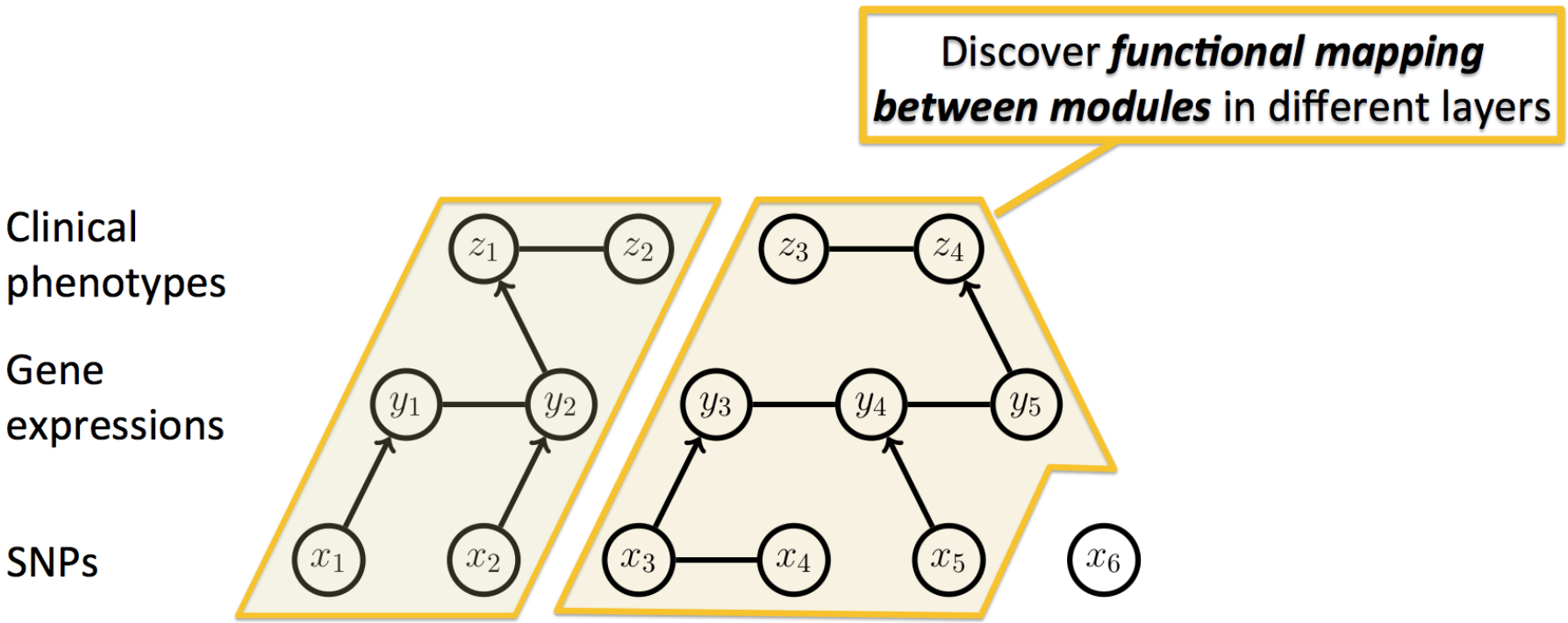
Learning Gaussian Chain Graphs: Our Approach

Given n samples stored in $\mathbf{X}$:

$$\min -L(\mathbf{X}; \mathbf{\Theta}, \mathbf{\Lambda}) + \lambda \sum_{\tau} \|\mathbf{\Theta}_{\tau,\mathrm{pa}(\tau)}\|_1 + \gamma \sum_{\tau} \|\mathbf{\Lambda}_{\tau}\|_1$$

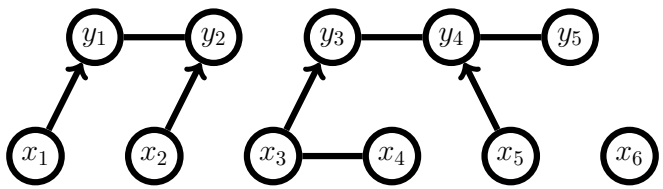
- Convex - global optimum
- Fast optimization algorithms (second half of talk!)

Structured Sparsity for Integrative Genomics



Structured Sparsity Recovery

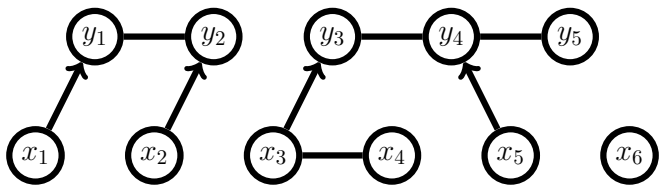
$$
\begin{aligned}
p(\mathbf{y}, \mathbf{x}) =& p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \\
=& \left(\exp(-\frac{1}{2}\mathbf{y}^T \mathbf{\Theta_{yy}} \mathbf{y} - \mathbf{x}^T \mathbf{\Theta_{xy}} \mathbf{y} / A_1(\mathbf{x})) \right) \left(\exp(-\frac{1}{2}\mathbf{x}^T \mathbf{\Theta_{xx}} \mathbf{x}) / A_2 \right)
\end{aligned}
$$



Structured Sparsity Recovery

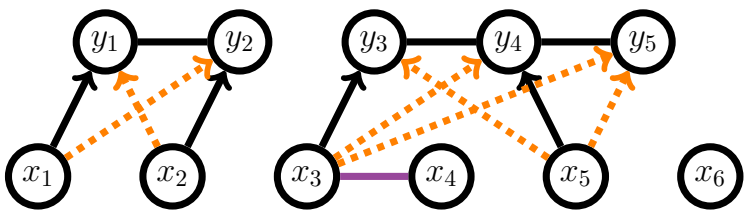
$$p(\mathbf{y},\mathbf{x}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$$

$$= \left(\exp(-\frac{1}{2}\mathbf{y}^T\mathbf{\Theta_{yy}}\mathbf{y} - \mathbf{x}^T\mathbf{\Theta_{xy}}\mathbf{y}/A_1(\mathbf{x})) \right) \left(\exp(-\frac{1}{2}\mathbf{x}^T\mathbf{\Theta_{xx}}\mathbf{x})/A_2 \right)$$



$$= N\left(-\mathbf{\Theta_{yy}}^{-1}\mathbf{\Theta_{xy}}\mathbf{x}, \mathbf{\Theta_{yy}}^{-1} \right) \left(\exp(-\frac{1}{2}\mathbf{x}^T\mathbf{\Theta_{xx}}\mathbf{x})/A_2 \right)$$

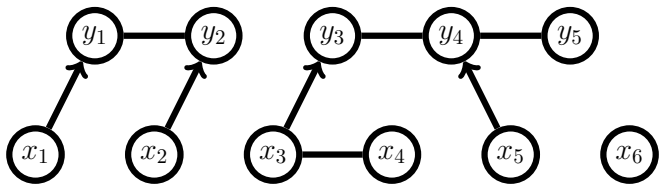
inference



Structured Sparsity Recovery

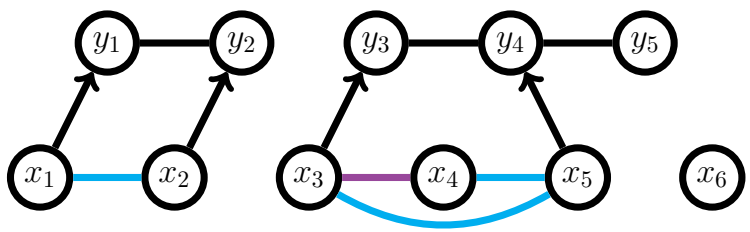
$$p(\mathbf{y},\mathbf{x}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$$

$$= \left(\exp(-\frac{1}{2}\mathbf{y}^T\mathbf{\Theta_{yy}}\mathbf{y} - \mathbf{x}^T\mathbf{\Theta_{xy}}\mathbf{y}/A_1(\mathbf{x})) \right) \left(\exp(-\frac{1}{2}\mathbf{x}^T\mathbf{\Theta_{xx}}\mathbf{x})/A_2 \right)$$

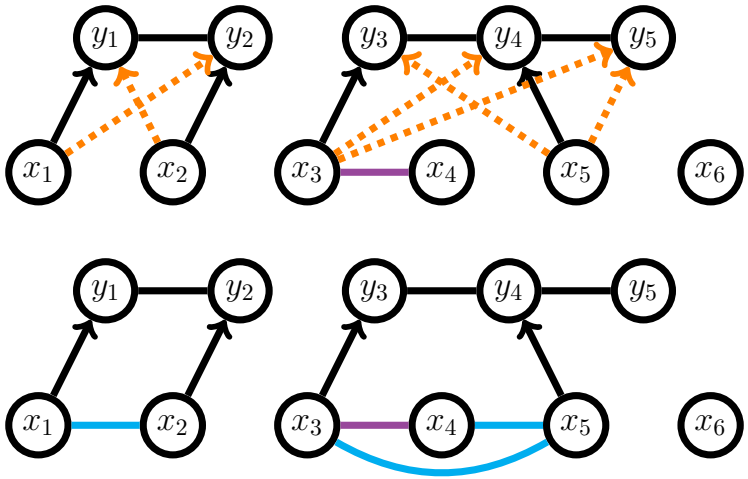


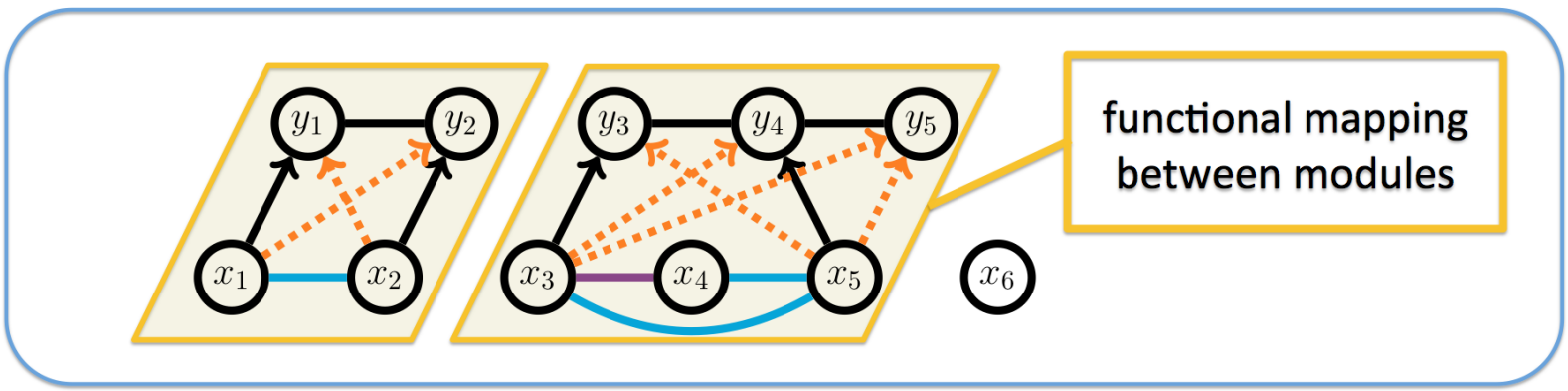
$$= N\left(\mathbf{0}, \begin{pmatrix} \mathbf{\Theta_{yy}} & \mathbf{\Theta_{xy}}^T \\ \mathbf{\Theta_{xy}} & \mathbf{\Theta_{xx}} + \mathbf{\Theta_{xy}}\mathbf{\Theta_{yy}}^{-1}\mathbf{\Theta_{xy}}^T \end{pmatrix}^{-1} \right)$$

moralization

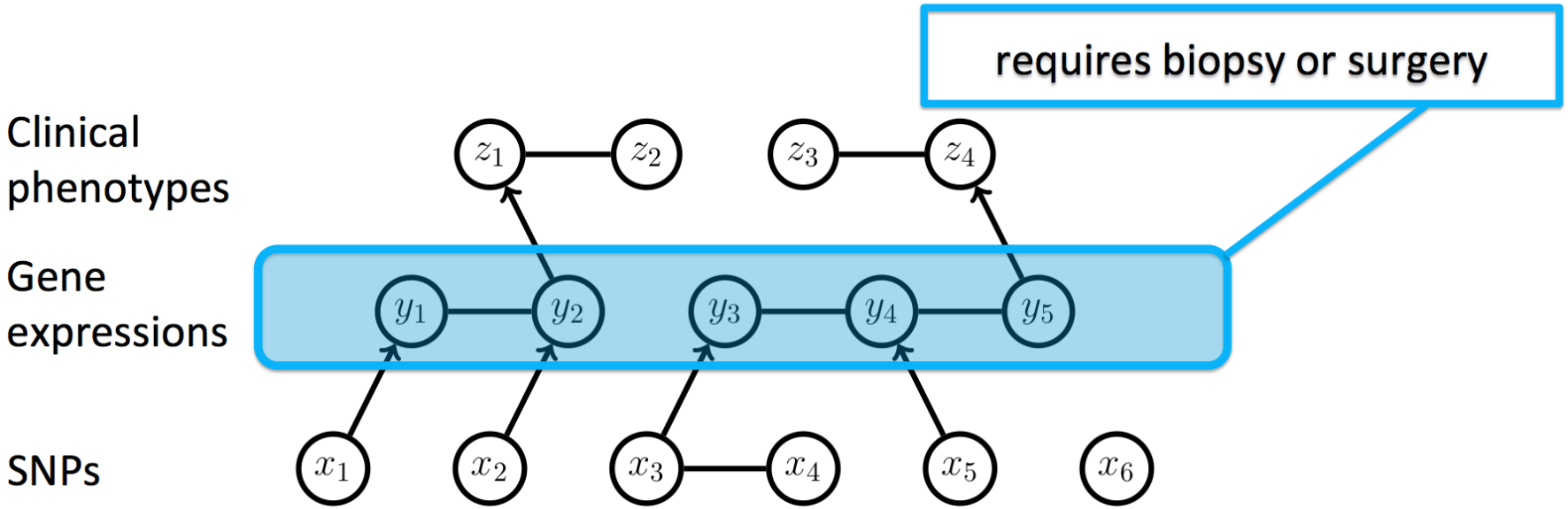


Structured Sparsity Recovery





Semi-supervised Learning



Fully observed data: $\mathcal{D}_o = \{\mathbf{X}_o, \mathbf{Y}_o, \mathbf{Z}_o\}$

Partially observed data: $\mathcal{D}_h = \{\mathbf{X}_h, \mathbf{Z}_h\}$

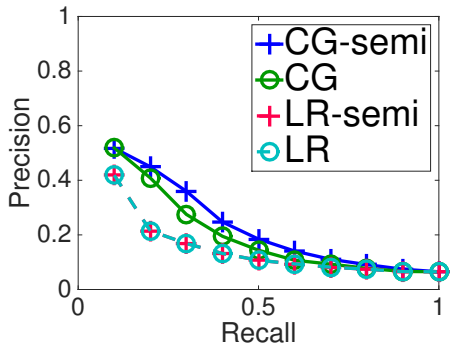
Maximize with EM algorithm:

$$\mathcal{L}(\mathcal{D}_o; \mathbf{\Theta}) + \mathrm{E}\big[\mathcal{L}(\mathcal{D}_h, \mathbf{Y}_h; \mathbf{\Theta})\big]$$

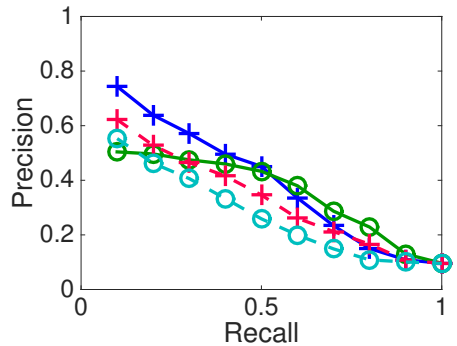
Results: Graph Structure Recovery

Sparse Linear Regression Ground Truth:

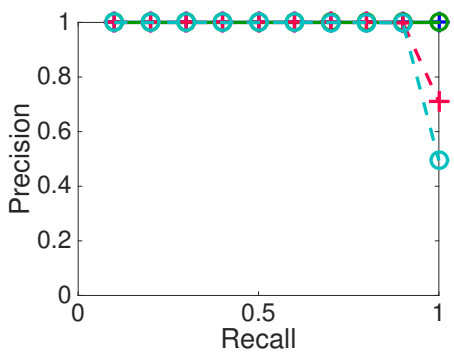
Θ_{yy}



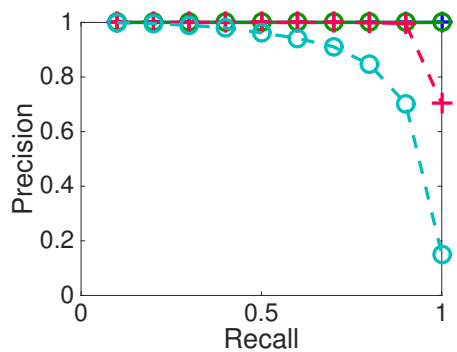
Θ_{zz}



B_{xy}



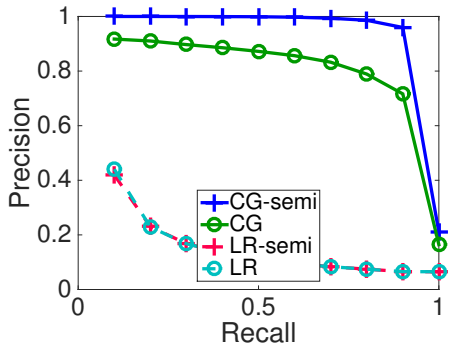
B_{yz}



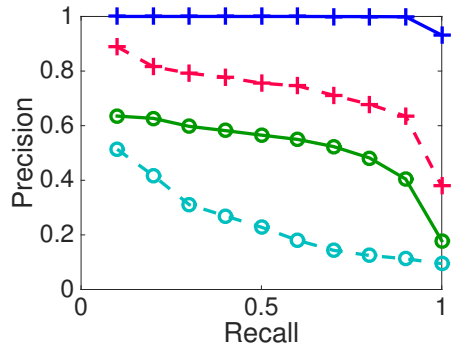
Results: Graph Structure Recovery

Sparse CGGM Ground Truth:

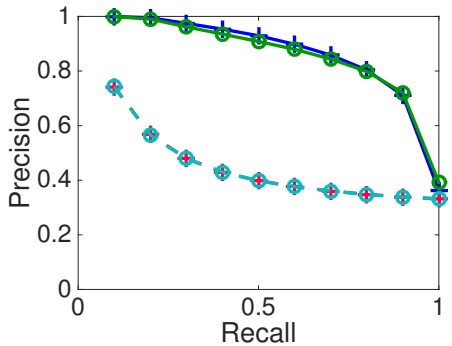
Θ_{yy}



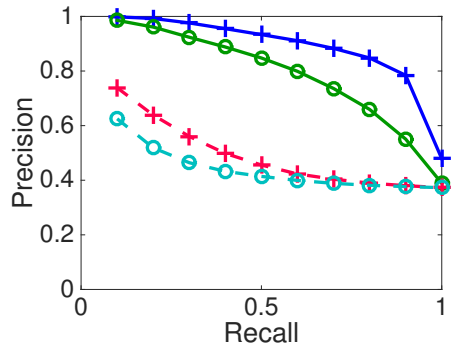
Θ_{zz}



B_{xy}



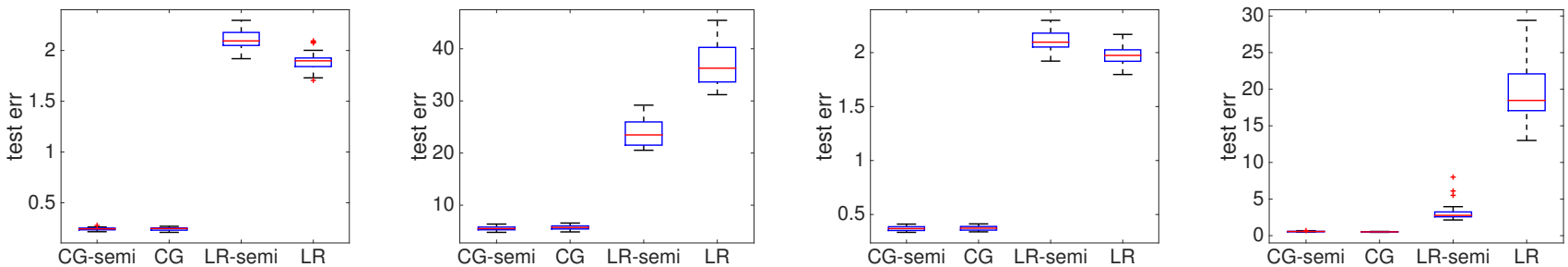
B_{yz}



Results: Prediction Tasks

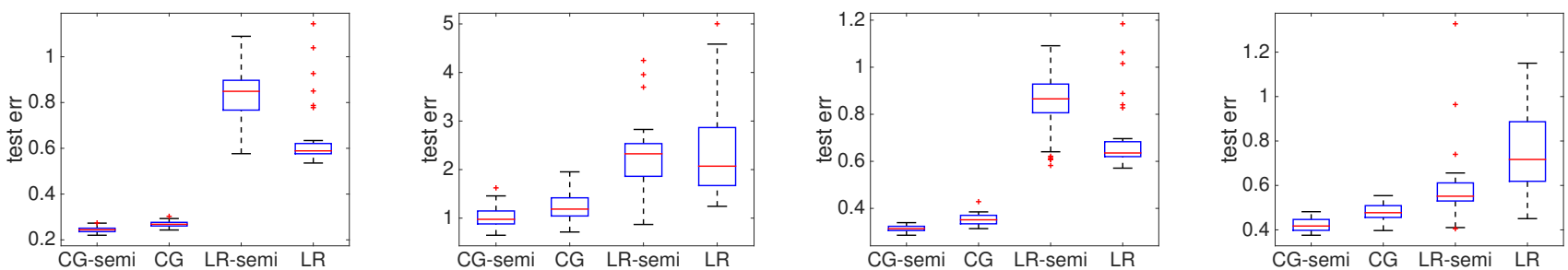
Sparse Linear Regression Ground Truth:

$\mathbf{y}|\mathbf{x},\mathbf{z}$ $\mathbf{z}|\mathbf{x}$ $\mathbf{y}|\mathbf{x}$ $\mathbf{z}|\mathbf{y}$

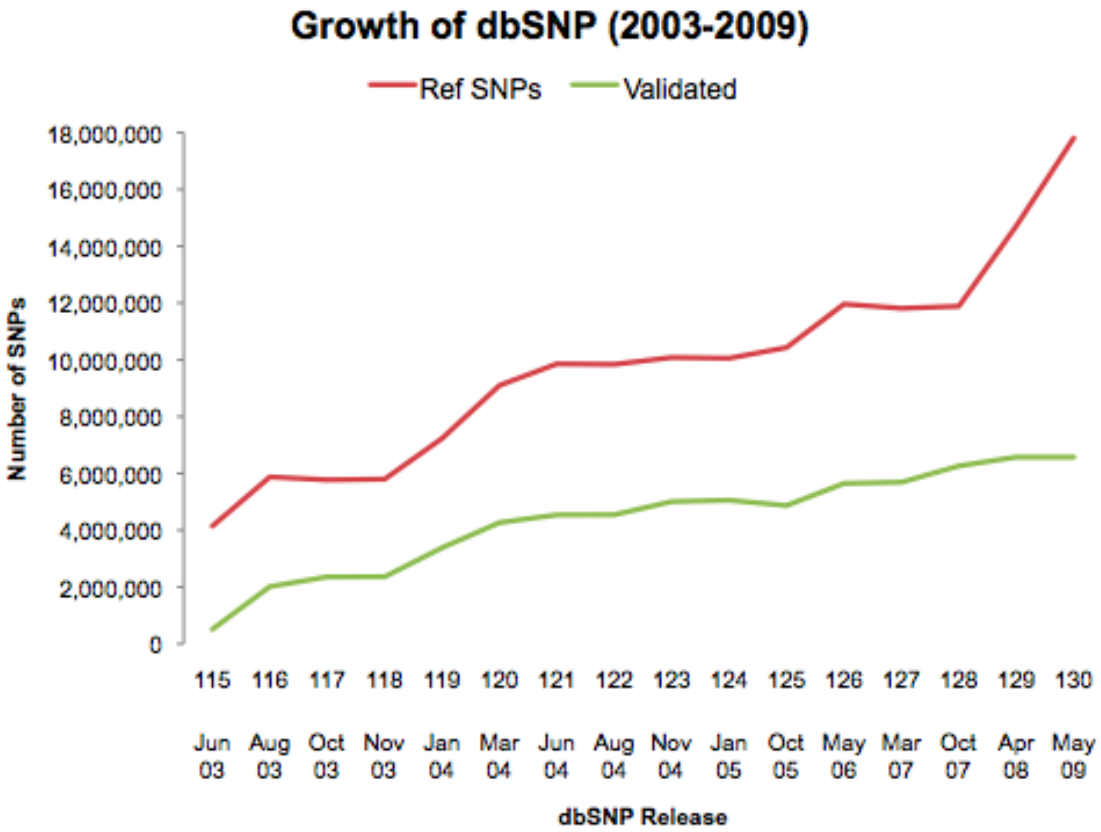


Sparse CGGM Ground Truth:

$\mathbf{y}|\mathbf{x},\mathbf{z}$ $\mathbf{z}|\mathbf{x}$ $\mathbf{y}|\mathbf{x}$ $\mathbf{z}|\mathbf{y}$



Scaling to High-Dimensional Datasets



Can we learn a model with a million SNPs?

Sparse CGGM Estimation

$$p(\mathbf{y}|\mathbf{x};\mathbf{\Lambda},\mathbf{\Theta}) = \exp\{-\mathbf{y}^T\mathbf{\Lambda}\mathbf{y} - 2\mathbf{x}^T\mathbf{\Theta}\mathbf{y}\}/Z(\mathbf{x}),$$

Applications in biology, energy forecasting, finance, etc

Sparse CGGM Estimation

$$p(\mathbf{y}|\mathbf{x};\mathbf{\Lambda},\mathbf{\Theta}) = \exp\{-\mathbf{y}^T\mathbf{\Lambda}\mathbf{y} - 2\mathbf{x}^T\mathbf{\Theta}\mathbf{y}\}/Z(\mathbf{x}),$$

Applications in biology, energy forecasting, finance, etc
Existing methods:

- OWL-QN (Sohn & Kim, 2012)
- FISTA (Yuan & Zhang, 2012)
- Proximal Newton Coordinate Descent (Wytock & Kolter, 2013)

Sparse CGGM Estimation: Optimization

Optimize over $\mathbf{\Lambda} \in \mathbb{S}_+^{q \times q}$ and $\mathbf{\Theta} \in \mathbb{R}^{p \times q}$:

$$\min_{\mathbf{\Lambda} \succ 0, \mathbf{\Theta}} f(\mathbf{\Lambda}, \mathbf{\Theta}) = g(\mathbf{\Lambda}, \mathbf{\Theta}) + h(\mathbf{\Lambda}, \mathbf{\Theta})$$

$g(\mathbf{\Lambda}, \mathbf{\Theta})$: smooth function from data log-likelihood
$h(\mathbf{\Lambda}, \mathbf{\Theta})$: non-smooth function for ℓ_1 penalty

Newton Coordinate Descent Method

$$\min_{\mathbf{\Lambda} \succ 0, \mathbf{\Theta}} f(\mathbf{\Lambda}, \mathbf{\Theta}) = g(\mathbf{\Lambda}, \mathbf{\Theta}) + h(\mathbf{\Lambda}, \mathbf{\Theta})$$

1. Find Generalized Newton direction:

$$\mathbf{D}_{\mathbf{\Lambda}}, \mathbf{D}_{\mathbf{\Theta}} = \operatorname*{argmin}_{\Delta_{\mathbf{\Lambda}}, \Delta_{\mathbf{\Theta}}} \bar{g}_{\mathbf{\Lambda}, \mathbf{\Theta}}(\Delta_{\mathbf{\Lambda}}, \Delta_{\mathbf{\Theta}}) + h(\mathbf{\Lambda} + \Delta_{\mathbf{\Lambda}}, \mathbf{\Theta} + \Delta_{\mathbf{\Theta}})$$

   - Precompute Hessian and gradient
   - Solve *Lasso* problem via coordinate descent over active set

2. Apply update with step size from line search

Newton Coordinate Descent: Scalability

Hessian has size $(q^2 + pq) \times (q^2 + pq)$

- Naive approach: compute $(q^2 + pq)^2$ elements

Newton Coordinate Descent: Scalability

Hessian has size $(q^2 + pq) \times (q^2 + pq)$

- Naive approach: compute $(q^2 + pq)^2$ elements
- Implicit Kronecker product: compute $q^2 + pq + p^2$ elements

Is this good enough?

Newton Coordinate Descent: Scalability

Hessian has size $(q^2 + pq) \times (q^2 + pq)$

- Naive approach: compute $(q^2 + pq)^2$ elements
- Implicit Kronecker product: compute $q^2 + pq + p^2$ elements

Is this good enough?

- Genomic dataset with $p = 34k, q = 10k$: > 50 hours
- Runs out of memory on 100Gb machine when $p + q > 80k$

Alternating Newton Coordinate Descent

Alternate between $\mathbf{\Lambda}$ and $\mathbf{\Theta}$

Updating $\mathbf{\Theta}$ given fixed $\mathbf{\Lambda}$ is a *Lasso* problem:
No quadratic approximation needed

- Precompute only $q^2 + p$ elements
- Avoid line search for $\mathbf{\Theta}$ update

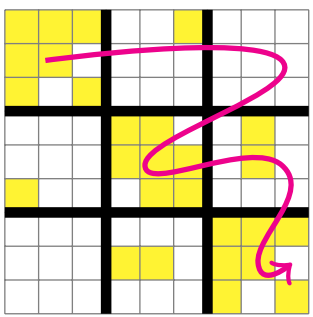
Still have memory problem...

Alternating Newton Block Coordinate Descent

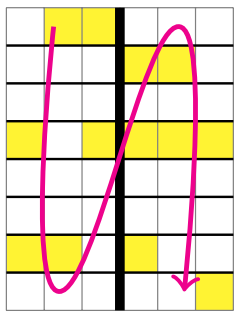
Partition Λ and Θ into blocks
For each block:

- Precompute Hessian and gradients needed within block
- Optimize within block via coordinate descent

Λ



Θ

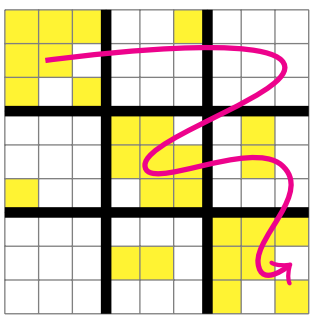


Alternating Newton Block Coordinate Descent

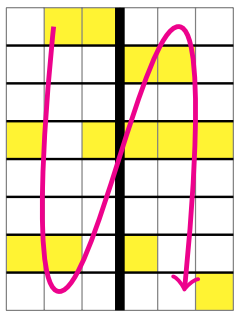
Partition Λ and Θ into blocks
For each block:

- Precompute Hessian and gradients needed within block
- Optimize within block via coordinate descent

Λ



Θ



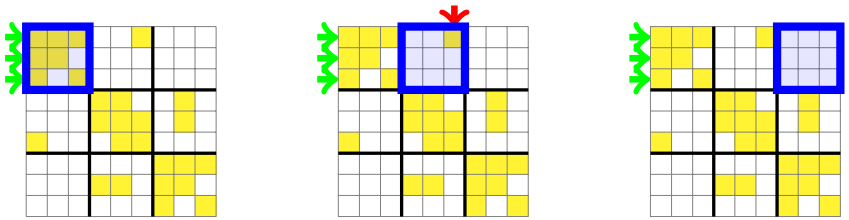
Idea: Choose partition to minimize duplicated work

Block Coordinate Descent for $\mathbf{\Lambda}$

- Partition $\mathbf{\Lambda}$ into $k \times k$ blocks
- Choose partition with graph clustering over $\mathbf{\Lambda}$

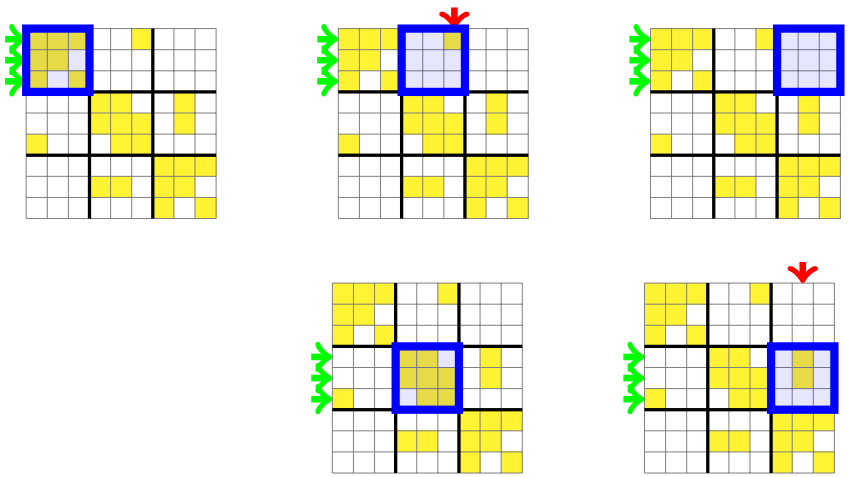
Block Coordinate Descent for $\mathbf{\Lambda}$

- Partition $\mathbf{\Lambda}$ into $k \times k$ blocks
- Choose partition with graph clustering over $\mathbf{\Lambda}$



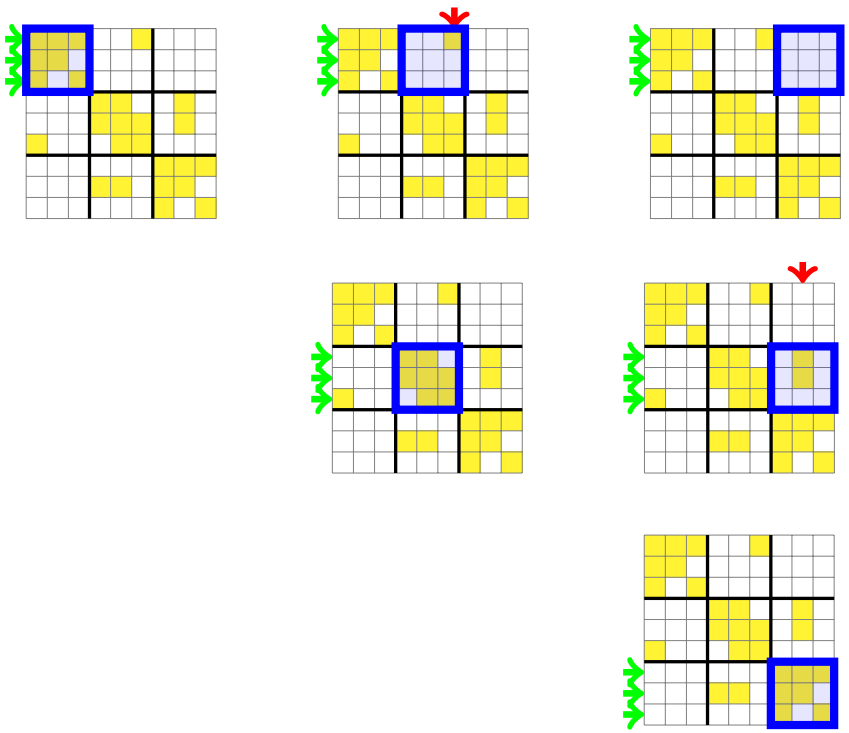
Block Coordinate Descent for $\mathbf{\Lambda}$

- Partition $\mathbf{\Lambda}$ into $k \times k$ blocks
- Choose partition with graph clustering over $\mathbf{\Lambda}$



Block Coordinate Descent for $\mathbf{\Lambda}$

- Partition $\mathbf{\Lambda}$ into $k \times k$ blocks
- Choose partition with graph clustering over $\mathbf{\Lambda}$

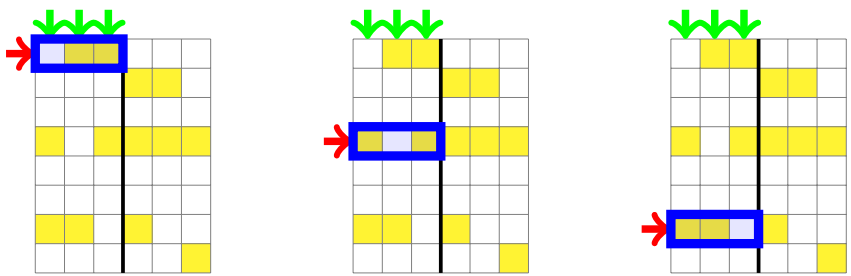


Block Coordinate Descent for $\mathbf{\Theta}$

- Partition $\mathbf{\Theta}$ into $p \times k$ blocks
- Choose partition with graph clustering over $\mathbf{\Theta}^T\mathbf{\Theta}$

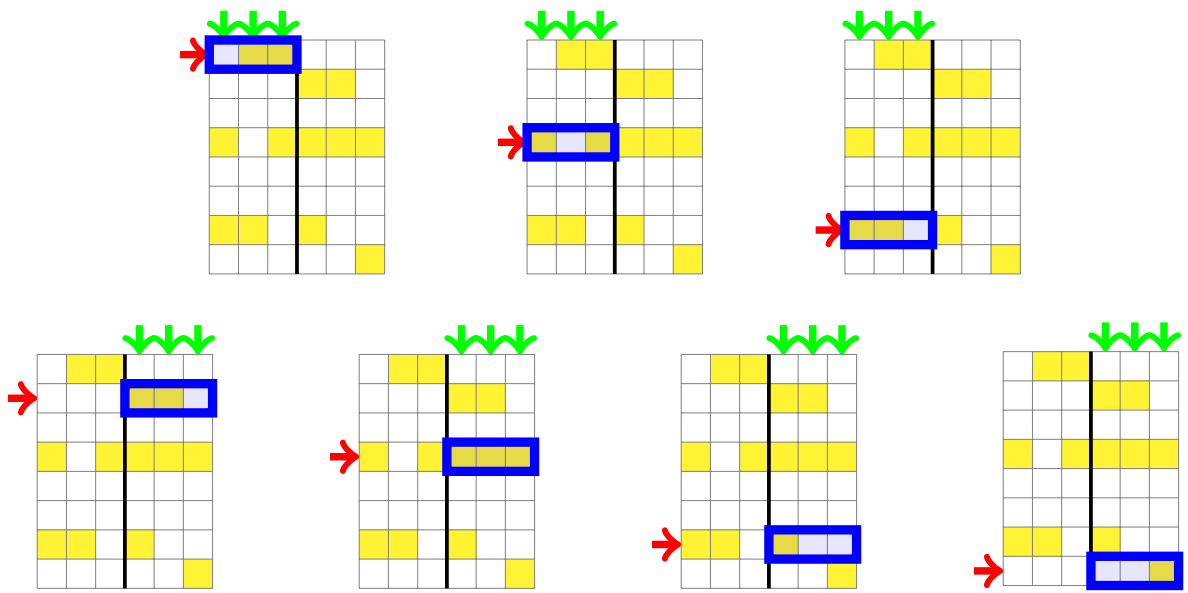
Block Coordinate Descent for $\boldsymbol{\Theta}$

- Partition $\boldsymbol{\Theta}$ into $p \times k$ blocks
- Choose partition with graph clustering over $\boldsymbol{\Theta}^T\boldsymbol{\Theta}$



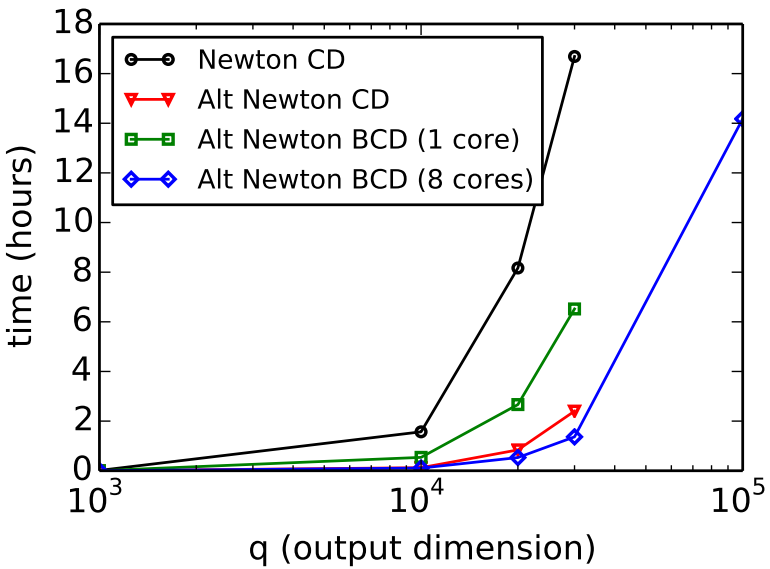
Block Coordinate Descent for $\boldsymbol{\Theta}$

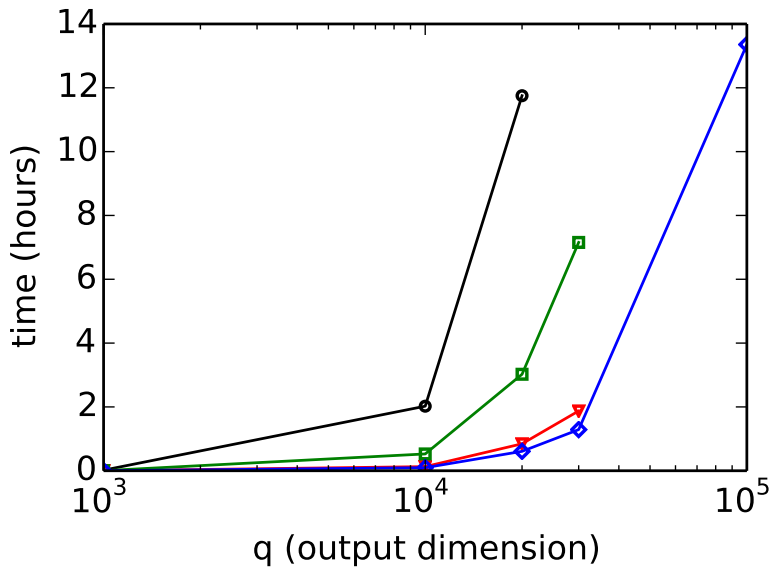
- Partition $\boldsymbol{\Theta}$ into $p \times k$ blocks
- Choose partition with graph clustering over $\boldsymbol{\Theta}^T\boldsymbol{\Theta}$



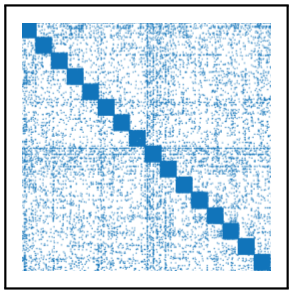
Results: Linear Graphs

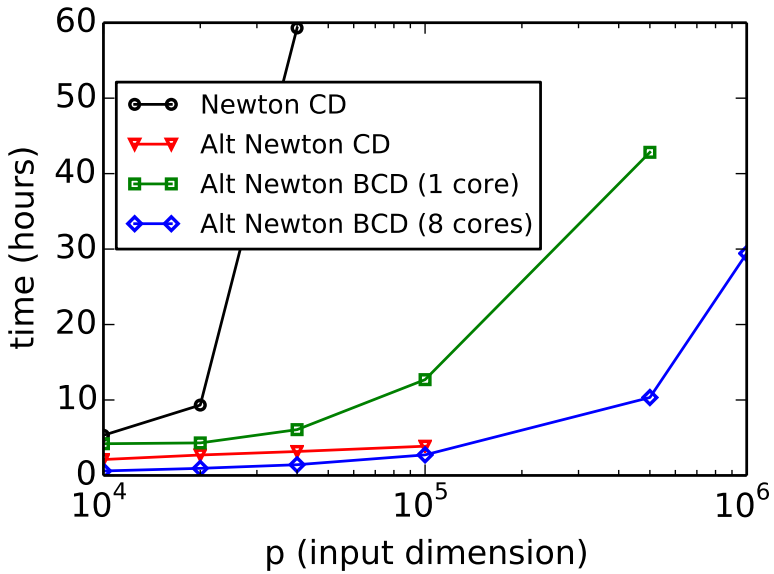
$$\mathbf{\Lambda}_{i,i-1} = 1, \mathbf{\Theta}_{i,i} = 1$$

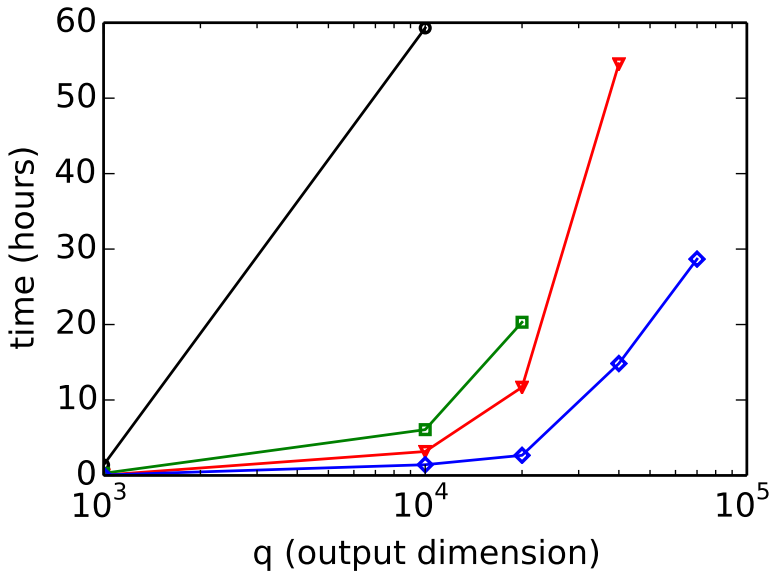




Results: Cluster Graphs

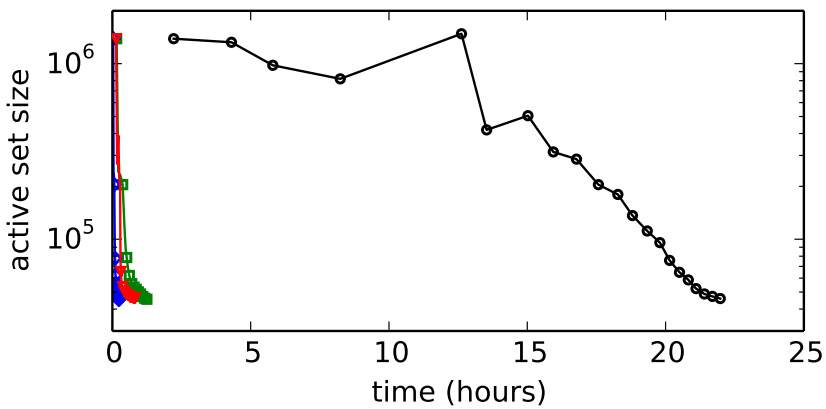


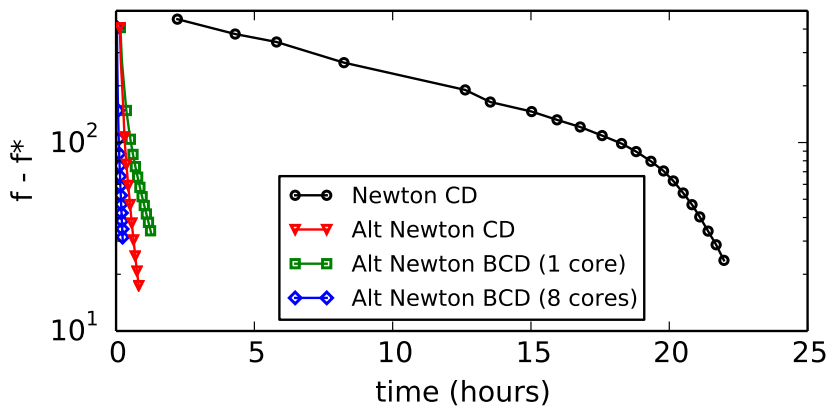




Results: Genome-wide Analysis

p	q	Newton CD	Alt Newton CD	Alt Newton BCD
34,249	3,268	22.0	0.51	**0.24**
34,249	10,256	> 50	2.4	**2.3**
442,440	3,268	*	*	**11**





Conclusions

Our approach:

- Learns structure within and across datasets
- Recovers structured sparsity
- Utilizes partially-available data
- Scales to millions of variables, billions of parameters

Thanks!