Inferring Gene Regulatory Networks
by Integrating Perturbation Screens and
Steady State Gene Expression Profiles

Ali Shojaie
Department of Biostatistics
University of Washington
ashojaie@uw.edu

IPM
December 26, 2011
Gene Regulatory Networks – Some Biology

Figure courtesy of Wikipedia
Features of Regulatory Networks:

• Sparse: on average \( \sim 2 \) TF's for each gene
• Directed: direction implies physical effect (causation)
• Cyclic: feedback loops
Gene Regulatory Networks

Features of Regulatory Networks:

- **Sparse**: on average \( \sim 2 \) TF’s for each gene
- **Directed**: direction implies physical effect (causation)
- **Cyclic**: feedback loops
Outline

1 Preliminaries

2 Proposed Approach

3 Results

4 Summary
Challenges in Estimating Gene Regulatory Networks

- Estimation of directed graphs is computationally \textbf{NP-hard}
- \textbf{Small sample size} $n$ compared to the dimension of the network $p$
- Directed edges present \textit{causality} which may not be determined from observational data
- There are few statistical models for analysis of (general) directed graphs with \textit{cycles} (with the exception of DBN, see S. and Michailidis 2010b for one such approach)
Current Approaches

- Using steady-state gene expression data:
  - Estimate the undirected association graphs
  - Estimate DAGs or equivalence class of DAGs

- Using perturbation screens, obtained by “perturbing” the biological system, often in the form of knockout or knockdown experiments, where in each experiment one or more genes are perturbed.
  - Model-based approaches: Nested Effect Models (NEM)
  - Heuristic approaches: Pinna et al (2010), PINNA
Current Approaches

- Using steady-state gene expression data:
  - Estimate the undirected association graphs
  - Estimate DAGs or equivalence class of DAGs

- Using perturbation screens, obtained by “perturbing” the biological system, often in the form of knockout or knockdown experiments, where in each experiment one or more genes are perturbed.
  - Model-based approaches: Nested Effect Models (NEM)
  - Heuristic approaches: Pinna et al (2010), PINNA
Undirected vs. Directed Graphs

- **Conditional independence relations** in directed graphs are represented by *d*-separation.

- Undirected conditional independence graphs are easy to estimate, and give useful information about dependencies among variables, but do not have information about direction of effect.

- Conditional independence graphs are equivalent to the *moral graph* of DAG and do not have the same structure.
Undirected vs. Directed Graphs

- **Conditional independence relations** in directed graphs are represented by *d-separation*

- Undirected conditional independence graphs are easy to estimate, and give useful information about dependencies among variables, but do not have information about direction of effect.

- Conditional independence graphs are equivalent to the *moral graph* of DAG and do not have the same structure.
Limitations of Available Data Sources

- Steady-state data are easy to obtain, but only represents association among genes and hence have insufficient informational content.

- On the other hand, perturbation data provide direct information on causal directions, but are expensive to obtain. This becomes more complicated if perturbing a particular gene is lethal.

- This data is obtained by “perturbing” the biological system, often in the form of knockout or knockdown experiments, where in each experiment one or more genes are knocked out/down. The data then measures the effect of the experiments on other genes in the network.

- In practice, due to limited sample size the perturbation data are often discretized: genes are categorized as up/down regulated or active/inactive.
Proposed Approach: Data Integration

Lemma (Rationale)

Discretized perturbation data

(i) *do not provide* enough information to *construct the structure of* regulatory networks.
Proposed Approach: **Data Integration**

**Lemma (Rationale)**

Discretized *perturbation data*

(i) *do not provide* enough information to *construct the structure* of regulatory networks.

(ii) *provide* enough information to *determine causal (topological) ordering(s)* of nodes.
Proposed Approach: **Data Integration**

**Lemma (Rationale)**

Discretized *perturbation data*

(i) *do not provide* enough information to *construct the structure of regulatory networks.*

(ii) *provide* enough information to *determine causal (topological) ordering(s) of nodes.*

**Proposed Method:**

I) Use perturbation data to determine *causal ordering(s)* among nodes

II) For each ordering from step (I), Use steady-state gene expression data to estimate the *structure* of the graph

III) Use model averaging to construct a *consensus graph*
Proposed Approach: The **RIPE** Algorithm

I) Determine causal orderings from perturbation screens

II) Use penalized likelihood to estimate a DAG for each ordering

III) Determine the "consensus" graph by model averaging

*: Regulatory Network Inference from joint Perturbation and Expression data
Step I) Determining Causal Orderings

- First, obtain the influence graph $P$ from the perturbation data (this can be done many different ways: p-value cutoff/fold-change cutoff etc)
Step I) Determining Causal Orderings

- First, obtain the influence graph $P$ from the perturbation data (this can be done many different ways: p-value cutoff/fold-change cutoff etc)
- In absence of noise, the influence graph is obtained from the original graph by connecting node $i$ to $j$ if there is a directed path from $i$ to $j$
Step I) Determining Causal Orderings

- First, obtain the influence graph \( P \) from the perturbation data (this can be done many different ways: p-value cutoff/fold-change cutoff etc)
- In absence of noise, the influence graph is obtained from the original graph by connecting node \( i \) to \( j \) if there is a directed path from \( i \) to \( j \)
- In practice, the influence graph will likely include false positive and false negative edges.
Step I) Determining Causal Orderings

- Create a hyper-graph of strong connected components (SCC), where each node is a collection of \( \geq 1 \) nodes that cannot be further ordered (i.e. there is a cycle).

- Find an ordering (topological sorting) of the SCC graph (note, this is by construction a DAG) using Depth First Search algorithm (DFS).

- Find all possible orderings of each connected component (using backtracking algorithm of Knuth, or Monte Carlo DFS MC-DFS).
Step I) Determining Causal Orderings

- Create a hyper-graph of strong connected components (SCC), where each node is a collection of \( \geq 1 \) nodes that cannot be further ordered (i.e. there is a cycle).

- Find an ordering (topological sorting) of the SCC graph (note, this is by construction a DAG) using Depth First Search algorithm (DFS).

- Find all possible orderings of each connected component (using backtracking algorithm of Knuth, or Monte Carlo DFS MC-DFS)

This is a relaxation of finding Hamiltonian paths on a graph and is closely related to TSP.
Step II) Estimation of the Structure

- Given a topological ordering of nodes, the nodes of the graph can be rearranged to form a DAG

- Causality and conditional dependence in DAGs are presented by structural equation models

\[ X_i = f_i(pa_i, \gamma_i), \quad i = 1, \ldots, p \]

where \( \gamma_i \) is the latent variable representing the pure effect of \( i \)-th variable

- For Gaussian random variables, \( f_i \) is linear

\[ X_i = \sum_{j \in pa_i} \rho_{ji}X_j + \gamma_i, \quad i = 1, \ldots, p \]
Step II) Estimation of the Structure

- Given a topological ordering of nodes, the nodes of the graph can be rearranged to form a DAG.

- Causality and conditional dependence in DAGs are presented by structural equation models:
  \[ X_i = f_i(pa_i, \gamma_i), \quad i = 1, \cdots, p \]
  where \( \gamma_i \) is the latent variable representing the pure effect of \( i \)-th variable.

- For Gaussian random variables, \( f_i \) is linear:
  \[ X_i = \sum_{j \in pa_i} \rho_{ji} X_j + \gamma_i, \quad i = 1, \cdots, p \]
Penalized likelihood estimation of DAGs

- Penalized likelihood estimation of the precision matrix:

\[
\hat{\Omega} = \arg\min_{\Omega \succ 0} \{-\log |\Omega| + \text{trace}(\Omega S) + \lambda J(\Omega)\}
\]

where \( S = n^{-1}X^TX \) is the empirical covariance matrix, \( J(\Omega) \) is the penalty and \( \lambda \) is the tuning parameter
Penalized likelihood estimation of DAGs

- Penalized likelihood estimation of the precision matrix:

\[
\hat{\Omega} = \arg\min_{\Omega \succ 0} \{ -\log |\Omega| + \text{trace} (\Omega S) + \lambda J(\Omega) \}
\]

where \( S = n^{-1}X^TX \) is the empirical covariance matrix, \( J(\Omega) \) is the penalty and \( \lambda \) is the tuning parameter.

- Given the ordering of nodes, it can be shown that (S. & Michailidis, 2010) the penalized likelihood problem can be solved directly as a function of the adjacency matrix \( A \in \mathcal{A} \)

\[
\hat{A} = \arg\min_{A \in \mathcal{A}} \{ \text{trace} \left[ (I - A)^T (I - A)S \right] + \lambda J(A) \}
\]

- Lasso (Tibshirani, 1996): \( J(A) = \sum |A_{ij}| \equiv \| A \|_1 \)

- Adaptive Lasso (Zou, 2006): \( J(A) = \sum W_{ij} |A_{ij}| \) with \( W_{ij} = |\tilde{A}_{ij}| - \gamma \), where \( \tilde{A} \) is the regular lasso estimate (\( \gamma = 1 \))

- Two-Stage Lasso (S. & Michailidis, 2010): \( W_{ij} = 1 \lor |\tilde{A}_{ij}| - 1 \); where \( \tilde{A} \) is the regular lasso estimate (\( \gamma = 1 \))
Penalized likelihood estimation of DAGs

- Penalized likelihood estimation of the precision matrix:

\[
\hat{\Omega} = \arg\min_{\Omega \succ 0} \{-\log |\Omega| + \text{trace}(\Omega S) + \lambda J(\Omega)\}
\]

where \( S = n^{-1}X^TX \) is the empirical covariance matrix, \( J(\Omega) \) is the penalty and \( \lambda \) is the tuning parameter.

- Given the ordering of nodes, it can be shown that (S. & Michailidis, 2010) the penalized likelihood problem can be solved directly as a function of the adjacency matrix \( A \in \mathcal{A} \)

\[
\hat{A} = \arg\min_{A \in \mathcal{A}} \{\text{trace}[(I - A)^T(I - A)S] + \lambda J(A)\}
\]

- Lasso (Tibshirani, 1996): \( J(A) = \sum |A_{ij}| \equiv \|A\|_1 \)
Penalized likelihood estimation of DAGs

- Penalized likelihood estimation of the precision matrix:
  \[
  \hat{\Omega} = \arg\min_{\Omega \succ 0} \{-\log |\Omega| + \text{trace}(\Omega S) + \lambda J(\Omega)\}
  \]
  where \( S = n^{-1}X^TX \) is the empirical covariance matrix, \( J(\Omega) \) is the penalty and \( \lambda \) is the tuning parameter

- Given the ordering of nodes, it can be shown that (S. & Michailidis, 2010) the penalized likelihood problem can be solved directly as a function of the adjacency matrix \( A \in \mathcal{A} \)
  \[
  \hat{A} = \arg\min_{A \in \mathcal{A}} \{\text{trace}[(I - A)^T(I - A)S] + \lambda J(A)\}
  \]

- Lasso (Tibshirani, 1996): \( J(A) = \sum |A_{ij}| \equiv \|A\|_1 \)

- Adaptive Lasso (Zou, 2006): \( J(A) = \sum W_{ij}|A_{ij}| \)
  \( W_{ij} = |\tilde{A}_{ij}|^{-\gamma} \), \( \tilde{A} \) a consistent estimate of \( A \)
Penalized likelihood estimation of DAGs

- Penalized likelihood estimation of the precision matrix:

\[
\hat{\Omega} = \arg\min_{\Omega \succ 0} \left\{ -\log |\Omega| + \text{trace} (\Omega S) + \lambda J(\Omega) \right\}
\]

where \( S = n^{-1}X^TX \) is the empirical covariance matrix, \( J(\Omega) \) is the penalty and \( \lambda \) is the tuning parameter.

- Given the ordering of nodes, it can be shown that (S. & Michailidis, 2010) the penalized likelihood problem can be solved directly as a function of the adjacency matrix \( A \in \mathcal{A} \):

\[
\hat{A} = \arg\min_{A \in \mathcal{A}} \left\{ \text{trace} \left[ (I - A)^T (I - A)S \right] + \lambda J(A) \right\}
\]

- **Lasso** (Tibshirani, 1996): \( J(A) = \sum |A_{ij}| \equiv \|A\|_1 \)

- **Adaptive Lasso** (Zou, 2006): \( J(A) = \sum W_{ij} |A_{ij}| \)

\[
W_{ij} = |\tilde{A}_{ij}|^{-\gamma}, \quad \tilde{A} \text{ a consistent estimate of } A
\]

- **Two-Stage Lasso** (S. & Michailidis, 2010)

\[
W_{ij} = 1 \lor |\tilde{A}_{ij}|^{-1}; \quad \tilde{A} \text{ is the regular lasso estimate (} \gamma = 1 \text{)}
\]
General Weighted Lasso Estimate of DAGs

- Let $A_+ = \max(A, 0)$ and $A_- = -\min(A, 0)$

$$\min_{A_+, A_- \geq 0} \text{trace} \left[ S(I - A_+ + A_-)^T (I - A_+ + A_-) + \lambda (A_+ + A_-) W + \Delta (A_+ + A_-) 1_{u^+} \right]$$

- here $W$ is matrix of weights for adaptive lasso, or ones for lasso, $\Delta$ is a large positive number and $1_{u^+}$ is the indicator matrix for upper triangular elements of a $p \times p$ matrix
General Weighted Lasso Estimate of DAGs

- Let \( A_+ = \max(A, 0) \) and \( A_- = -\min(A, 0) \)

\[
\min_{A_+, A_- \geq 0} \text{trace} \left[ S(I - A_+ + A_-)^T (I - A_+ + A_-) + \lambda (A_+ + A_-) W + \Delta (A_+ + A_-) 1_{u^+} \right]
\]

- here \( W \) is matrix of weights for adaptive lasso, or ones for lasso, \( \Delta \) is a large positive number and \( 1_{u^+} \) is the indicator matrix for upper triangular elements of a \( p \times p \) matrix

- With the ordering of variables known, the general weighted lasso penalty problem can be reformulated as \( p - 1 \) \( \ell_1 \)-regularized least squares problems:

\[
\hat{A}_{k,k-1} = \arg\min_{\theta \in \mathbb{R}^{k-1}} \left\{ n^{-1} \| X_{n,k-1} \theta - X_{n,k} \|_2^2 + \lambda \sum_{j=1}^{k-1} | \theta_j | w_j \right\}, \quad k = 2, \ldots, p
\]

where \( l = j : 1 \leq j \leq l \).
### Computational Complexity

- Using **coordinate descent** algorithm, cost of penalized likelihood method is \( O(np^2) \).

- For sparse graphs, computational cost of **PC-Algorithm** is \( O(p^q) \), where \( q \) is the maximal neighborhood size.

<table>
<thead>
<tr>
<th>CPU time</th>
<th>pcalg</th>
<th>lasso</th>
<th>alasso</th>
</tr>
</thead>
<tbody>
<tr>
<td>p=100, n=100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p=100, n=1000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p=1000, n=100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p=1000, n=1000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Computational Complexity

- Using coordinate descent algorithm, cost of penalized likelihood method is $O(np^2)$

- For sparse graphs, computational cost of PC-Algorithm is $O(p^q)$, where $q$ is the maximal neighborhood size

![Graph showing CPU time for different algorithms and parameter settings](image-url)
Step III) Building a Consensus Graph

- For each ordering, the estimated graph is a DAG
- However, the true graph may include cycles. Also, results from one ordering may be inaccurate (noise...)
  - $L_q$: lower $q$th quantile of the (penalized) negative log-likelihoods
  - $Q = \{ o \in \mathcal{O} : \ell(o) \leq L_q \}$ set of orderings for these likelihoods

$$\hat{A}_{i,j}^c = \frac{1}{|Q|} \sum_{k \in Q} 1_{\{|A_{i,j}^k| > 0\}}$$

$$\hat{E} = \{(i,j) : \hat{A}_{i,j}^c \geq \tau\}$$
Step III) Building a Consensus Graph

- For each ordering, the estimated graph is a DAG
- However, the true graph may include cycles. Also, results from one ordering may be inaccurate (noise...)
  - $L_q$: lower $q$th quantile of the (penalized) negative log-likelihoods
  - $Q = \{ o \in \mathcal{O} : \ell(o) \leq L_q \}$ set of orderings for these likelihoods
  - $\hat{A}_{i,j}^c = \frac{1}{|Q|} \sum_{k \in Q} 1\{|A_{i,j}^k| > 0\}$
  - $\hat{E} = \{(i,j) : \hat{A}_{i,j}^c \geq \tau\}$
Some Comments

- When the perturbation data includes cycles, the consensus graph will be cyclic.

- It can be shown that under certain regulatory conditions, as the number of perturbation and steady-state expression samples increase, the RIPE is consistent in high dimensional sparse settings. More specifically, we can show that (in high dimensional sparse settings),
  
  (i) With probability converging to 1, the edges of the regulatory network are correctly estimated.

  (ii) With probability converging to 1, the signs of the regulatory effects (repression/inhibition) are correctly estimated.
Simulate Network: DAG of size $p = 20$
Data Generation

Perturbation data: Adjacency matrices of true and noisy influence graphs

Steady-state expression data: generated $n = 50$ Gaussian observations according to the true DAG.
Comparison of $F_1$ measures
How Many Orderings Do We Need?

For $P_3$, there are a total of 3962 orderings using the backtracking algorithm.
A More Complicated Example: DREAM-4 Challenge

- The DREAM project (Dialogue for Reverse Engineering Assessments and Methods) is an attempt to construct realistic regulatory networks.
- DREAM-4 challenge had multiple competitions, including reverse engineering 5 networks of size 100 selected from true regulatory components of yeast and E-coli.
- The perturbation data is simulated based on the true network (using coupled ODE).
- Two types of perturbation data are available: knockout and knockdown experiments.
- The DREAM does not provide iid samples (it includes perturbation and time series data), so we simulated $n = 100$ samples from Gaussian distribution.
- The algorithm of Pinna et al (PINNA) was the winner of the high dimensional reconstruction challenge (on networks of size 100).
DREAM Network 1 (Simplest)
DREAM Network 5 (Most Difficult!)
Comparison of $F_1$ Measures
269-node Network of Yeast Transcription Factors

- **Perturbation data**: knockout experiments

- **Steady-state expression data**: $n = 200$ day-to-day variation samples of yeast (publicly available), not really iid!!

- Used 10,000 orderings

- To evaluate: use available data on yeast regulatory network, which is (most likely) incomplete. Therefore, “false positives” may be true edges
269-node Network of Yeast Transcription Factors

- Significance of true positives (TP), in comparison to the BioGrid network
- Histograms show number of TP’s in random networks of equal sizes
High Dimensional Cyclic Graphs ($p = 1000$)

Effect of FP and FN errors

![Graph showing the effect of FP and FN errors on F1 scores for different algorithms and error rates.]

- PCALG
- FFLDR
- RIPE−1000

- **FP:** 0.02% (200), 0.04% (400), 0.06% (600)
- **FN:** 2% (200), 4% (400), 6% (600)
- **FP&FN:** 0.01% FN:1% (200), 0.02% FN:2% (400), 0.03% FN:3% (600)

- **F1** scores are plotted for each combination of error levels, showing the degradation in performance with increasing error rates.

- Key points:
  - No error (no err)
  - FP errors
  - FN errors
  - FP&FN errors
Summary

- Estimation of regulatory networks is difficult!!
- Available data differ in informational content and available sample size (and hence noise level)
- **Penalized likelihood methods** provide an efficient solution for estimation of DAGs when variables inherit natural ordering
- The proposed RIPE algorithm efficiently integrates two data sources for estimation of regulatory networks
- Improved performance especially when good quality steady-state data is available
- Transformed a computationally hard problem to solving many easy problems (**parallelizable**)
Thank You!
I) Estimation of Causal Orderings: some details

Although DFS finds an ordering of a DAG in $O(p + |E|)$ time, however, finding all orderings in a cyclic graph is NP-hard. On the other hand, when the influence graph has cycles, DFS does not give all the orderings. Consider the following simple graphs:

**LEFT:** DFS (starting at 1) gives: $1 \prec 2 \prec 4 \prec 3$ and $1 \prec 4 \prec 2 \prec 3$

**RIGHT:** DFS (starting at 1) gives: $1 \prec 2 \prec 3 \prec 4$. However, $2 \prec 3 \prec 4 \prec 1$, $3 \prec 4 \prec 1 \prec 2$ and $4 \prec 1 \prec 2 \prec 3$ are all valid orderings.

To find all the orderings for a strong component, we adapt the Knuth’s backtracking algorithm (Knuth & Szwarcfiter, 1974). The backtracking algorithm is NP-hard, so in practice, we combine backtracking (for small super nodes) with a Monte-Carlo (sampling-based) version of DFS (for larger super nodes).
Choosing the P-value Cutoff
Knockout data from DREAM: Network 1

- Size of largest connected component
- Number of edges
- P-value cut-off

![Graph showing the relationship between size of largest connected component and number of edges for different P-value cutoffs.](image-url)
Choosing the P-value Cutoff
Knockout data from DREAM: Network 5
Choice of $q$ and $\tau$
Consistency

Theorem

Let $s$ be the total number of true edges in the graph. Suppose that for some $a > 0$, $p = p(n) = O(n^a)$ and $|\text{pa}_i| = O(n^b)$, where $sn^{2b-1}\log n = o(1)$ as $n \to \infty$. Moreover, suppose that there exists $\nu > 0$ such that for all $n \in \mathbb{N}$ and all $i \in V$, $\text{Var}(X_i | X_{1:i-1}) \geq \nu$ and there exists $\delta > 0$ and some $\xi > b$ such that for every $i \in V$ and for every $j \in \text{pa}_i$, $|\pi_{ij}| \geq \delta n^{-(1-\xi)/2}$, where $\pi_{ij}$ is the partial correlation between $X_i$ and $X_j$ after removing the effect of the remaining variables.

Assume that $\lambda \asymp dn^{-(1-\xi)/2}$ for some $b < \zeta < \xi$ and $d > 0$. Then, the RIPE estimate obtained with an adaptive lasso penalty in the second stage of the algorithm, with initial weights found using RIPE with lasso penalty using a coefficient satisfying $\lambda^0 = O(\sqrt{\log p/n})$ satisfies the following

(i) With probability converging to 1, the edges of the regulatory network are correctly estimated.

(ii) With probability converging to 1, the signs of the regulatory effects (repression/inhibition) are correctly estimated.