Computational methods for inferring gene regulatory networks: Part 2

Sushmita Roy
sroy@biostat.wisc.edu
STAT/BMI 877: Statistical Methods for Molecular Biology
https://www.biostat.wisc.edu/~kendzior/STAT877/

Feb 23rd 2017
Goals for this lecture

• Per-module based network inference methods
• Methods for integrative regulatory network inference
• Prior-based data integration
  – Integrating structure priors with Bayesian networks
  – Integrating parameter priors with Dependency networks
Two classes of expression-based methods

• Per-gene/direct methods

• Module based methods
Per-module methods

- Find regulators for an entire module
  - Assume genes in the same module have the same regulators
- Module Networks (Segal et al. 2005)
- Stochastic LeMoNe (Joshi et al. 2008)
Module Networks

• Motivation:
  – Most complex systems have too many variables
  – Not enough data to robustly learn networks
  – Large networks are hard to interpret

• Key idea: Group similarly behaving variables into “modules” and learn the same parents and parameters for each module

• Relevance to gene regulatory networks
  – Genes that are co-expressed are likely regulated in similar ways

Segal et al 2005, JMLR
A regulatory module: set of genes with similar regulatory state

Experimental conditions

<table>
<thead>
<tr>
<th>Condition</th>
<th>Genesregulated by TF A</th>
<th>Genesregulated by TF B</th>
<th>Genesregulated by TF C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zn Pho</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MMS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H2O2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DTT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AA starv</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N starv</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C sources</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Regulatory gene modules

Gasch & Eisen, 2002
Definition of a module

- Statistical definition (specific to module networks by Segal 2005)
  - A set of random variables that share a statistical model
- Biological definition of a module
  - Set of genes that are co-expressed and co-regulated
Modeling questions in Module Networks

• How to model the CPD between parent and children?
  – Regression Tree

• How to learn module networks?
Defining a Module Network

• Denoted by $\mathcal{M} = \{ \mathcal{A}, \mathcal{S}, \mathcal{\Theta} \}$
• $\mathcal{S}$: Structure, specifying the parents of each module
• $\mathcal{A}$: Assignment of $X_i$ to module $k$, $\mathcal{A}(X_i) = k$
• $\mathcal{\Theta}$: Parameterizing CPD $P(M_j|Pa_{Mj})$, $Pa_{Mj}$ are parents of module $M_j$
  -- Each variable $X_i$ in $M_j$ has the same conditional distribution
Bayesian network vs Module network

• Bayesian network
  – CPD per random variable
  – Learning only requires to search for parents

• Module network
  – CPD per module
  – Learning requires parent search and module membership assignment
Bayesian network vs Module network

(a) Bayesian network

(b) Module network

Each variable takes three values: UP, DOWN, SAME
Learning a Module Network

• Given
  – training dataset $D = \{x^1, \cdots, x^m\}$
  – fixed number of modules: $K$

• Learn
  – Module assignments of each variable
  – CPD structure and parameters of each module
Module network learning algorithm

Input:

\( D \) // Data set

\( K \) // Number of modules

Output:

\( M \) // A module network

Learn-Module-Network

\( \mathcal{A}_0 = \text{cluster } \mathcal{X} \text{ into } K \text{ modules} \)

\( S_0 = \text{empty structure} \)

Loop \( t = 1, 2, \ldots \) until convergence

\( S_t = \text{Greedy-Structure-Search}(\mathcal{A}_{t-1}, S_{t-1}) \)

\( \mathcal{A}_t = \text{Sequential-Update}(\mathcal{A}_{t-1}, S_t); \)

Return \( M = (\mathcal{A}_t, S_t) \)
Initial modules identified by expression clustering
Iterations in learning Module Networks

Learn regulators/CPD per module

Revisit the modules

Module \( M_1 \) and \( M_3 \) get updated
Modeling questions in Module Networks

• How to score and learn module networks?
• How to model the CPD between parent and children?
  – Regression Tree
Regression tree to capture CPD in the Module network

Each path captures a mode of regulation of $X_3$ by $X_1$ and $X_2$

Expression of target modeled using Gaussians at each leaf node
Assessing the value of using Module Networks

- Using simulated data
  - Generate data from a known module network
  - Known module network was in turn learned from real data
    - 10 modules, 500 variables
  - Evaluate using
    - Test data likelihood
    - Recovery of true parent-child relationships in learned module network
- Using real gene expression data
  - External validation of modules (Gene ontology, motif enrichment)
  - Cross-check with literature
Test data likelihood

Each line type represents size of training data

10 Modules is the best for almost all training data set sizes

Figure 7: Performance of learning from synthetic data as a function of the number of modules and training set size. The x-axis corresponds to the number of modules, each curve corresponds to a different number of training instances, and each point shows the mean and standard deviations from the 10 sampled data sets. (a) Log-likelihood per instance assigned to held-out data. (b) Average score per instance on the training data.

We first evaluated the generalization to unseen test data, measuring the likelihood ascribed by the learned model to 4500 unseen instances. The results, summarized in Figure 7(a), show that, for all training set sizes, except the smallest one with 25 instances, the model with 10 modules performs the best. As expected, models learned with larger training sets do better; but, when run using the correct number of 10 modules, the gain of increasing the number of data instances beyond 100 samples is small and beyond 200 samples is negligible.
Recovery of graph structure

Figure 8: (a) Fraction of variables assigned to the 10 largest modules. (b) Average percentage of correct parent-child relationships recovered (fraction of parent-child relationships in the true model recovered in the learned model) when learning from synthetic data for models with various number of modules and different training set sizes. The x-axis corresponds to the number of modules, each curve corresponds to a different number of training instances, and each point shows the mean and standard deviations from the 10 sampled data sets.

To test whether we can use the score of the model to select the number of modules, we also plotted the score of the learned model on the training data (Figure 7(b)). As can be seen, when the number of instances is small (25 or 50), the model with 10 modules achieves the highest score and for a larger number of instances, the score does not improve when increasing the number of modules beyond 10. Thus, these results suggest that we can select the number of modules by choosing the model with the smallest number of modules from among the highest scoring models.

A closer examination of the learned models reveals that, in many cases, they are almost a 10-module network. As shown in Figure 8(a), models learned using 100, 200, or 500 instances and up to 50 modules assigned $\geq 80\%$ of the variables to 10 modules. Indeed, these models achieved high performance in Figure 7(a). However, models learned with a larger number of modules had a wider spread for the assignments of variables to modules and consequently achieved poor performance.

Finally, we evaluated the model's ability to recover the correct dependencies. The total number of parent-child relationships in the generating model was 2250. For each model learned, we report the fraction of correct parent-child relationships it contains. As shown in Figure 8(b), our procedure recovers 74\% of the true relationships when learning from a data set with 500 instances. Once again, we see that, as the variables begin fragmenting over a large number of modules, the learned structure contains many spurious relationships. Thus, our results suggest that, in domains with a modular structure, statistical noise is likely to prevent overly detailed learned models such as Bayesian networks from extracting the commonality between different variables with a shared behavior.
Application of Module networks to yeast expression data

Regulator selection → Data selection

Pre-processing

Candidate regulators → Expression data

Regulation program learning

Clustering

Gene partition → Gene reassignment to modules

Functional modules

Module network procedure

Motifs

Graphical presentation

Hypotheses & validation

Annotations

Motifs search → Annotation analysis

Post-processing

 Segal et al, Regev, Pe’er, Gasch 2005
Modeling the relationship between regulators and targets

- suppose we have a set of (8) genes that all have in their upstream regions the same activator/repressor binding sites
The Respiration and Carbon Module

Regression tree representing rules of regulation
Global View of Modules

- modules for common processes often share common
  - regulators
  - binding site motifs
Take away points

- Network inference from expression provides a promising approach to identify cellular networks
- Graphical models are one representation of networks that have a probabilistic and graphical component
  - Network inference naturally translates to learning problems in these models
- Different types of algorithms to learn these networks
  - Per-gene methods
    - Sparse Candidate, GENIE3: learn regulators for individual genes
  - Per-module methods
    - Module networks: learn regulators for sets of genes/modules
    - Methods that combine per-gene and per-module
- Network reconstruction from expression alone is challenging
  - New methods aim to combine other types of measurements (next)
Goals for this lecture

• Per-module based network inference methods
• Methods for integrative regulatory network inference
• Prior-based data integration
  – Integrating structure priors with Bayesian networks
  – Integrating parameter priors with Dependency networks
Regulation of gene expression happens at multiple levels.

Joseph R. Ecker, Wendy A. Bickmore, Inês Barroso, Jonathan K. Pritchard, Yoav Gilad & Eran Segal
Nature 489, 52–55 (06 September 2012) doi:10.1038/489052a
Types of data for reconstructing transcriptional networks

- **Expression data**
  - Genome-wide mRNA levels from multiple microarray or RNA-seq experiments
  - Gene expression can come from time courses as well as single time point

- **Complementary datasets**
  - ChIP-chip and ChIP-seq
  - Sequence specific motifs
Classes of methods for integrative unsupervised network inference

• Simple aggregation of different networks
  – Flynnet (Marbach & Roy et al., Genome Research 2012)

• Prior-based approaches
  – Parameter prior based approaches
    • Inferelator (Greenfield et al., Bioinformatics 2013)
    • Lirnet (Lee et al., Plos Genetics 2009)
  – Structure prior based approaches
    • Dynamic Bayesian networks
      – Hill et al., Bioinformatics, 2012
      – Werhli and Dirk Husmeier, Statistical Applications in Genetics and Molecular Biology, 2007
    • MERLIN-P, Siahpirani & Roy, Nucleic Acids Research, 2016

• Other
  – iRafNet (Petralia et al., Bioinformatics, 2015)
Goals for this lecture

• Per-module based network inference methods
• Methods for integrative regulatory network inference
• Prior-based data integration
  – Integrating structure priors with Bayesian networks
  – Integrating parameter priors with Dependency networks
Prior-based approaches for network inference

- Given
  - Gene expression data and
  - Complementary data that supports the presence of an edge
    - Presence of a sequence motif on a gene promoter
    - ChIP-chip/seq binding of factor X on gene Y’s promoter
- Do
  - Predict which regulators drive the expression of a target gene
- How?
  - Place a prior on the graph where the prior is obtained from complementary data
Bayesian formulation of network inference

\[ P(G|D) \propto P(D|G)P(G) \]

Posterior distribution  Data likelihood  Model prior

Optimize posterior distribution of graph given data
Energy function of a network $G$

- A function that measures agreement between a given graph $G$ and prior knowledge
- Allows one to incorporate both positive and negative prior knowledge
Energy function on a graph

- A graph $G$ is represented by a binary adjacency matrix
  - $G_{ij} = 0$ if there is no edge from node $i$ to node $j$
  - $G_{ij} = 1$ if there is an edge from $i$ to $j$
  - $G_{ji} = 1$ if there is an edge from $j$ to $i$
- Encode a “prior” network as follows:
  - $B_{ij} = 0.5$ if we don’t have any prior
  - $0 < B_{ij} < 0.5$ if we know that there is no edge
  - $B_{ij} > 0.5$ if we know there is an edge
- Energy of $G$ is
  \[
  E(G) = \sum_{ij=1} |B_{ij} - G_{ij}|
  \]
Energy function of a graph

• Energy $E$ of a network $G$ is defined as

\[ E(G) = \sum_{ij=1} |B_{ij} - G_{ij}| \]

• This is 0 when there is perfect agreement between prior knowledge $B$ and $G$

• Higher the energy of $G$ the greater the mismatch
Using the energy to define a prior distribution of a graph

- A prior distribution for a graph $G$ can be defined using $E(G)$

$$P(G|\beta) = \frac{1}{Z(\beta)} \exp(-\beta E(G))$$

- This is also called a Gibbs distribution
- $\beta$ is the hyperparameter: parameter of the prior distribution

- $Z$ is the partition function

$$Z(\beta) = \sum_{G} \exp(-\beta E(G))$$

- In general the partition function is hard to compute
Incorporating multiple sources of prior networks

- Suppose we have two sources of prior networks
- We can represent them as two prior networks $B^1$ and $B^2$
- And define the energy of $G$ with respect to both of these

\[
E_1(G) = \sum_{i,j=1}^{i,j} |B_{i,j}^1 - G_{ij}|
\]

\[
E_2(G) = \sum_{i,j=1}^{i,j} |B_{i,j}^2 - G_{ij}|
\]
Prior distribution incorporating multiple prior networks

• The prior takes the form of another Gibbs distribution

\[
P(G|\beta_1, \beta_2) = \frac{1}{Z(\beta_1, \beta_2)} \exp\left(-\left(\beta_1 E_1(G) + \beta_2 E_2(G)\right)\right)
\]

• This can be extended to more prior networks in the same way

• The partition functions are in general hard to compute

• However, for a particular class of BNs, these partition functions can be computed easily

• This class is called Dynamic Bayesian Networks
Dynamic Bayesian networks

• Bayesian networks that we have seen so far do not allow for cyclic dependencies
• If we have time series data, we can overcome this limitation using a Dynamic Bayesian network
Dynamic Bayesian Networks

- A Dynamic Bayesian network (DBN) is a Bayesian network that can model temporal/sequential data
- DBN is a Bayes net for dynamic processes
- A DBN also has a graph structure and conditional probability distributions
- The DBN specifies how observations at a future time point may arise from previous time points.
Assume we have a time course with $T$ time points specifying activity of $p$ different variables.

Let $X^t = \{X^t_1, \ldots, X^t_p\}$ denote the set of random variables at time $t$.

A DBN over these variables defines the joint distribution of $P(X)$, where $X = \{X^1, \ldots, X^T\}$.

A DBN, like a BN, has a directed acyclic graph $G$ and parameters $\Theta$.

G typically specifies the dependencies between time points.

- In addition we need to specify dependence (if any) at $t=0$. 

**Notation**
A DBN for $p$ variables and $T$ time points

Dependency at the first time point

$X^2$: Variables at time $t=2$
Stationary assumption in a Bayesian network

The stationarity assumption states that the dependency structure and parameters do not change with $t$

$$P(X^{t+1} | X^t) = P(X^t | X^{t-1})$$

Due to this assumption, we only need to specify dependencies between two sets of variables
Dynamic Bayesian networks

Joint Probability Distribution can be factored into a product of conditional distributions:

\[ P(X|G, \Theta) = P(X^1) \prod_{i=1}^{p} \prod_{t=2}^{T} P(X^t_i | X^t_{\pi_G(i)}, \theta_i) \]

Graph encoding dependency structure

Parents of \( X^t_i \) defined by the graph

DBNs make the assumption of stationarity
The partition function for a prior over DBN

- In the DBN, suppose we allow parents only from the previous time point
- We allow each node to have at most $m$ parents
- The prior distribution decomposes over individual nodes and their possible parent sets

$$E(G) = \sum_{i=1}^{N} \mathcal{E}(n, \pi_G(n))$$

Parents of $n$ in graph $G$

$$\mathcal{E}(n, \pi_G(i)) = \sum_{j \in \pi_G(n)} (1 - B_{jn}) + \sum_{j \notin \pi_G(n)} B_{jn}$$
The partition function for a DBN prior

- The partition function is computed easily by summing over all variables and their potential parent sets

\[ Z(\beta) = \sum G \exp(-\beta E(G)) \]

\[ = \sum_{\pi_1} \cdots \sum_{\pi_N} \exp(-\beta (E(1, \pi_1) + \cdots , + E(N, \pi_N))) \]

\[ = \prod_{i=1}^{N} \sum_{\pi_i} \exp(-\beta (E(i, \pi_i))) \]

Each summation represents a sum over possible configurations for the parent set. If we restrict the number of parents to \( m \), this is polynomial in \( N \).
Learning problems in DBNs

• Parameter learning: Given known temporal dependencies between random variables estimate the parameters from observed measurements

• Structure learning: Given data, learn both the graph structure and parameters
  – Complexity of learning depends upon the order of the model

• We will look at an example of learning DBNs in the presence of prior data
Bayesian Inference of Signaling Network Topology in a Cancer Cell Line (Hill et al 2012)

- Protein signaling networks are important for many cellular diseases
  - The networks can differ between normal and disease cell types
- But the structure of the network remains incomplete
- Temporal activity of interesting proteins can be measured over time, that can be used infer the network structure
- Build on prior knowledge of signaling networks to learn a better, predictive network
- BNs are limiting because they do not model time
Integrating prior signaling network into the DBN

- A Bayesian approach to graph learning

\[ P(G|\mathbf{D}) \propto P(\mathbf{D}|G)P(G) \]

Data likelihood \hspace{1cm} Graph prior

- Graph prior is encoded as a special type of Gibbs distribution (Following Mukherjee & Speed 2008)

\[ P(G) \propto \exp(\lambda f(G)) \]

Prior strength \hspace{1cm} Graph features

- Where \( f(G) = -|E(G)\setminus E^*| \) is defined as the number of edges in the graph \( G \), that are not in the prior set \( E^* \)

- This prior is the extreme case of B, where we set \( B_{ij} = 0 \) if there is no edge and \( B_{ij} = 1 \) if there is an edge in the prior network
Calculating posterior probabilities of edges

• For each edge $e$, we need to calculate

$$P(e \mid D) = \sum_{G \in G} \mathbb{I}_{e \in G} P(G \mid D)$$

• Although this is intractable in general, this work makes some assumptions
  – Allow edges only forward in time
    • The learning problem decomposes to smaller per-variable problems that can be solved by variable selection
  – Assume $P(G)$ factorizes over individual edges
  – To compute the posterior probability, the sum goes over all possible parent sets
    • Assume a node can have no more than $d_{\text{max}}$ parents
Results on simulated data

20 variables, 4 time-courses
8 time points

Prior network had 54 extra edges and did not have 10 of the ground truth edges
Applying DBNs to infer signaling network topology

Hill et al., Bioinformatics 2012
Application of DBNs to signaling networks

• Dataset description
  – Phospho-protein levels of 20 proteins
  – Eight time points
• Use known signaling information as priors in the model
• Estimate CPDs as conditional regularized Gaussians
• Assume a first-order Markov model
  – $X^t$ depends on $X^{t-1}$
Inferred signaling network using a DBN

Figure S3: Prior network. Existing biology is captured and integrated during modeling using a prior probability distribution on graphs

\[ P(G) \propto \exp(\lambda f(G)) \]

where \( f(G) = -|E(G)\setminus E^\ast| \)

\( E(G) \) is the set of edges contained in \( G \) and \( E^\ast \) is a set of a priori expected edges. The graph shows edge set \( E^\ast \). Edges represent interactions through time. Each node also has a self-loop edge (i.e. an edge starting and finishing at the same node, these are not displayed). The edge set includes expected indirect edges which operate via components not included in our study.

Prior also had self-loops that are not shown

Inferred signaling network

Prior network
Using the DBN to make predictions

• Although many edges were expected, several edges were unexpected
• Select novel edges based on posterior probability and test them based on inhibitors
• For example, if an edge was observed from $X$ to $Y$, inhibition of $X$ should affect the value of $Y$, if $X$ is a regulator of $Y$
• Example edges tested
  – MAPKp to STAT3p(S727) with high probability (0.98)
    • Apply MEKi, which is an inhibitor of MAPK, and measure MAPKp and STAT3p post inhibition
  – AKTp to p70S6Kp, AKTp to MEKp and AKTp to cJUNp
Experimental validation of links

Add MAPK inhibitor and measure MAPK and STAT3

MAPK is significantly inhibited (P-value 5X10^{-4})
STAT3 is also inhibited (P-value 3.3X10^{-4})

Their success is measured by the difference in the levels of the targets as a function of the levels of the inhibitors
Summary

• Prior knowledge can be incorporated as an energy function on a graph and used to define a prior distribution
  – Extensible to multiple priors
• We saw an example of using priors with a Dynamic Bayesian network that helped to improve the quality of the inferred network.
• The DBN was also used to make predictions that were validated experimentally.
Goals for this lecture

• Per-module based network inference methods
• Methods for integrative regulatory network inference
• Prior-based data integration
  – Integrating structure priors with Bayesian networks
  – Integrating parameter priors with Dependency networks
Inferelator approach to integrate additional data

• Extend the a previous approach (Inferelator) to integrate an existing prior
  ‒ A network inference method that can incorporate both time course and single time point data

• Two approaches to integrate prior graph structure
  ‒ Modified Elastic Net (MEN)
  ‒ Bayesian Best Subset Regression (BBSR)

• Both approaches rely on a linear regression model
Modeling the relationship between regulator and target

- Time series

\[ y_i(t_{k+m}) = \sum_{p \in P_i} \beta_{i,p} x_p(t_k) \]
\[ i = 1, \ldots, N, \quad k = 1, \ldots, K - 1 \]

- Steady state

\[ x_i(e_l) = \tau_i \sum_{p \in P_i} \beta_{i,p} x_p(e_l), \]
\[ i = 1, \ldots, N, \quad l = 1, \ldots, L \]

Number of genes \quad Number of samples

Network inference: Estimate coefficients \( \beta_{i,p} \)
Modified Elastic Net (MEN)

- Elastic Net regression

Minimize sum of squared error

\[ \mathcal{E}_i(\beta) = \sum_{r=1}^{R} \left( y_i(r) - \sum_{p \in P_i} \beta_{i,p} x_p(r) \right)^2 \]

Subject to

L1 norm \hspace{1cm} L2 norm

\[(1 - \xi) \sum_{p \in P_i} |\beta_{i,p}| + \xi \sum_{p \in P_i} \beta_{i,p}^2 \leq s_i \sum_{p \in P_i} |\beta_{i,p}^{ols}| \]

Estimate via cross validation
MEN continued

• The modification to Elastic net

\[(1 - \xi) \sum_{p \in P_i} |\theta_{i,p} \beta_{i,p}| + \xi \sum_{p \in P_i} \beta_{i,p}^2 \leq s_i \sum_{p \in P_i} |\beta_{i,p}^{ols}|\]

Set this <1 so that if there is a prior edge between \(x_p \rightarrow y_i\), the regression coefficient will be penalized less.
Bayesian Best Subset Regression (BBSR)

- Based on a Bayesian framework of model selection
  - Search among all subsets of regulators and pick the best one to minimize trade off between data fit and model complexity
- Assume that the expression level $y$ is distributed according to a Gaussian distribution

$$
(y | \beta, \sigma^2, X) \propto N_n(X\beta, \sigma^2 I)
$$

- Place a prior distribution on the parameters (beta), and incorporate prior knowledge of interactions in the parameters

$$
\rho(\beta | \sigma^2) \propto N_n(\beta^0, g(X'X)^{-1} \sigma^2),
$$

$$
g \in (0, \infty)
$$
\( \rho(\beta|\sigma^2) \propto N_n(\beta^0, g(X'X)^{-1}\sigma^2), \)

\( g \in (0, \infty) \)

- \( g \) can be tuned to provide a trade-off between the prior \( \beta_0 \) and the OLS solution
- When \( g \) is larger, \( \beta \) is closer to the OLS solution
- When \( g \) is smaller, \( \beta \) is closer to the prior
- Inferelator uses a \( p \)-dimensional vector \( \bar{g} \) for \( p \) predictors

\[ \bar{g} = \left\{ g, \cdots, \frac{1}{g}, g, \cdots, \frac{1}{g} \right\}, \text{where} \ g > 1 \]
BBSR continued

• The posterior distribution over the parameters is given as:

\[ p(\beta|y, \sigma^2) \propto N\left( \frac{\beta^0}{g+1} + \frac{g}{g+1} \beta^{OLS}, \sigma\left( \frac{g}{g+1} (X'X)^{-1} \right) \right) \]

• \( g \) can be tuned to provide a trade-off between the prior and the OLS solution

• When \( g \) is larger, beta is closer to the OLS solution

• When it is smaller, beta is closer to the prior

• The prior is set to be a vector of all 0s
BBSR continued

- Inferelator uses a $p$-dimensional vector for $p$ predictors

\[ \bar{g} = \left\{ g, \cdots, \frac{1}{g}, g, \cdots, \frac{1}{g} \right\}, \text{where } g > 1 \]

Predictors with prior are set to $g$ (push more towards the OLS solution)
BBSR model selection

• The final step in BBSR is to determine the best model out of $2^p$ possible sets
• $p$ cannot be very high: the approach sets $p$ to 10
• The best model is the one that minimizes prediction error and has the lowest model complexity
Experimental setup

• Three datasets
  – DREAM4: In silico dataset with 100 nodes
  – *E. coli* dataset from DREAM5
  – *B. subtilis* dataset

• Evaluation based on AUPR
  – Ranking of edges obtained from a bootstrapping strategy
Workflow of experiments

Fig. 1. Method flow chart. Our method takes as input an expression dataset. To build a mechanistic model of gene expression, we create time-lagged response and design variables, such that the expression of the TF is time-lagged with respect to the expression of the target. We then resample the response and designing matrices, running model selection (using either MEN or BBSR) for each resample. This generates an ensemble of networks, which we rank combine into one final network.
How does the prior parameter affect the performance?

Fig. 2. Effect of weight parameter on performance. We use all GSIs as the set of PKIs, and evaluate performance (in terms of AUPR) against the set of GSIs. We evaluate this performance for a variety of choices of the weight parameter for both methods.
Ability to recover new edges is not hampered on adding prior

Fig. 4. Performance change on the leave-out set. PKIs were sampled randomly from 20%, 40%, 60% and 80% of the GSIs in five repetitions. We defined the leave-out set as the set of GSIs that are not PKIs. Here, we compare the AUPR of the leave-out set when using PKIs (y-axis) to the AUPR when not using PKIs (x-axis). Points above the line indicate a performance increase when PKIs are used.
What happens when one adds noisy priors?

Fig. 5. Robustness to incorrect prior information. For each dataset, we considered half of the GSIs as TPIs, and added varying numbers of FPIs that were not GSIs. We show the AUPR of both methods for multiple choices of the respective weight parameters, as well as methods that do not use any PKIs (horizontal lines). Additionally, we show the performance of a naive interaction ranking method, which places all PKIs at the top of the list (gray bars).

Low and high in BBSR and MEN means more sparse or less sparse
Inferelator key points

• Based on linear regression models
• Handles time series and steady state data
• Prior is incorporated at the edge weight
  – Modified Elastic Net aims to reduce the penalty for edges with prior support
  – BBSR aims to put a prior distribution on regression weights
Take away points

- We have seen different ways to incorporate other data types to improve the quality of the inferred network
- Bayesian networks with structure prior
  - Use an energy function to assess concordance
  - Sensitive to incorrect prior information
- Dependency networks with priors
  - Linear regression approach aims to reduce the penalty on inferred edges
- Several other expression-based methods have been extended to integrate other types of information
  - Dependency networks
    - MERLIN-P (Siahpirani & Roy, NAR 2016)
    - GENIE3->iRafNet
    - ModuleNetworks-> Physical Module networks (we did not see this)
Summary of integrative inference for regulatory networks

• Several expression-based methods have been extended to integrate other types of information
  – Bayesian networks
  – Dependency networks
    • Inferelator
    • MERLIN-P
  – GENIE3->iRafNet
  – ModuleNetworks-> Physical Module networks (we did not see this)
• Graph priors at the parameter or structure level can be informative
References

• Reconstructing Gene Regulatory Networks with Bayesian Networks by Combining Expression Data with Multiple Sources of Prior Knowledge. Adriano V. Werhli and Dirk Husmeier 2007


