Graph structure learning for network inference: Module Networks

Sushmita Roy
sroy@biostat.wisc.edu
Computational Network Biology
Biostatistics & Medical Informatics 826
Computer Sciences 838
https://compnetbiocourse.discovery.wisc.edu
Sep 20th 2016

Some of the material covered in this lecture is adapted from BMI 576
Goals for this lecture

• Per-module network inference methods
• Module networks
• A few case studies of Module networks
• Combining per-gene and per-module network inference methods (if time permits)
Two classes of expression-based network inference methods

• Per-gene/direct methods

• Module based methods (This lecture)
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
Regulatory gene modules

A regulatory module: set of genes with similar regulatory state

Experimental conditions

Genes

Genes

Modules

High expression

Low expression

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
Bayesian network vs Module network

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

• Module network
  – CPD per module
    • Same CPD for all random variables in the same 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
Modeling questions in Module Networks

• How to score and learn module networks?
• How to model the CPD between parent and children?
  – Regression Tree
Defining a Module Network

• A probabilistic graphical model over $N$ random variables $X = \{X_1, \cdots, X_N\}$

• Set of module variables $M_1, \ldots, M_K$

• Module assignments $A$ that specifies the module (1-to-K) for each $X_i$

• CPD per module $P(M_j|Pa_{M_j})$, $Pa_{M_j}$ are parents of module $M_j$
  – Each variable $X_i$ in $M_j$ has the same conditional distribution
Learning a Module Network

- Given training dataset \( D = \{x^1, \ldots, x^m\} \), fixed number of modules \((K)\)

- Learn
  - Module assignments \( A \) of each variable to a module
  - The parents of each module given by structure \( S \)
Score of a module network

- Module network makes use of a Bayesian score

\[ P(S, A \mid D) \propto P(A)P(S \mid A)P(D \mid S, A) \]

\[ \text{score}(S, A : D) = \]

\[ \log P(A) + \log P(S \mid A) + \log P(D \mid S, A). \]
Score of a module network continued

\[ \log P(D|S, A) = \log \int P(D|S, A, \theta) P(\theta|S, A) d\theta \]

Integrate parameters out

Decomposes over each module

\[ \log \prod_{j=1}^{k} \int L_j(U, X, \theta_{M_j}|U : D) P(\theta_{M_j}|U) d\theta_{M_j}|U \]

Decomposes over each module

\[ \sum_{j=1}^{K} \log \int L_j(U, X, \theta_{M_j}|U : D) P(\theta_{M_j}|U) d\theta_{M_j}|U \]

\(U\): Set of parents defined by \(S\)

\(X\): Set of variables.

For computing each \(L_j\) term we would need only the variables parents associated with module \(j\)
Defining the data likelihood

\[ X^j = \{ X_i \in X \mid A(X_i) = j \} \]

Likelihood of module \( j \)  
\[ = \prod_{j=1}^{K} L_j(\mathbf{Pa}_{M_j}, X^j, \theta_j : \mathcal{D}) \]

\[ L_j = \prod_{m=1}^{\mathcal{D}} \prod_{X_i \in X^j} P(x_i[m] | \mathbf{pa}_{M_j}[m], \theta_j) \]

\( K \): number of modules, \( X^j \): \( j^{th} \) module \hspace{1em} \( \mathbf{Pa}_{M_j} \): Parents of module \( M_j \)
Data likelihood example

\[ S(M_2, MSFT) = \hat{S}(AMAT, MSFT) + \hat{S}(MOT, MSFT) + \hat{S}(INTL, MSFT) \]

\[ S(M_3, AMAT, INTL) = \hat{S}(DELL, AMAT, INTL) + \hat{S}(HPQ, AMAT, INTL) + \]

If we are learning conditional probability distributions from the exponential family (e.g., discrete distribution, Gaussian distributions, and many others), then the local likelihood functions can be reformulated in terms of sufficient statistics of the data. The sufficient statistics summarize the relevant aspects of the data. Their use here is similar to that in Bayesian networks (Heckerman, 1998), with one key difference. In a module network, all of the variables in the same module share the same parameters. Thus, we pool all of the data from the variables in \( X_j \), and calculate our statistics based on this pooled data. More precisely, let \( S_j(M_j, Pa_{M_j}) \) be a sufficient statistic function for the CPD \( P(M_j | Pa_{M_j}) \). Then the value of the statistic on the data set \( D \) is

\[ \hat{S}_j = \sum_{m=1}^{M} \sum_{x_i \in X_j} S_j(x_i[m], Pa_{M_j}[m], \theta_{M_j | Pa_{M_j}}) \]
Module network learning algorithm

Input:

\( D \) // Data set
\( K \) // Number of modules

Output:

\( M \) // A module network

Learn-Module-Network

\( \mathcal{A}_0 = \) cluster \( \mathcal{X} \) into \( K \) modules
\( S_0 = \) empty structure

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

\( S_t = \) Greedy-Structure-Search\( (\mathcal{A}_{t-1}, S_{t-1}) \)
\( \mathcal{A}_t = \) 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
Module assignment search

• Happens in two places
• Module initialization
  – Interpret as clustering of the random variables
    \[ X = \{ X_1, \ldots, X_N \} \]
• Module re-assignment
Module initialization as clustering of variables

(a) Data

(b) Standard clustering

(c) Initialization

for module network
Module re-assignment

• Must preserve the acyclic graph structure
• Must improve score
• Module re-assignment happens using a **sequential update** procedure:
  – Update only one variable at a time
  – The change in score of moving a variable from one module to another while keeping the other variables fixed
Module re-assignment via sequential update

Input:
- $D$ // Data set
- $A_0$ // Initial assignment function
- $S$ // Given dependency structure

Output:
- $A$ // improved assignment function

Sequential-Update

$A = A_0$

Loop

For $i = 1$ to $n$

For $j = 1$ to $K$

$A' = A$ except that $A'(X_i) = j$

If $\langle G_M, A' \rangle$ is cyclic, continue

If $\text{score}(S, A' : D) > \text{score}(S, A : D)$

$A = A'$

Until no reassignments to any of $X_1, \ldots X_n$

Return $A$
Modeling questions in Module Networks

• How to score and learn module networks?
• How to model the CPD between parent and children?
  – Regression Tree
Representing the Conditional probability distribution

- $X_i$ are continuous variables
- How to represent the distribution of $X_i$ given the state of its parents?
- How to capture context-specific dependencies?
- Module networks use a regression tree
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
A regression tree

- A rooted binary tree $T$
- Each node in the tree is either an interior node or a leaf node
- Interior nodes are labeled with a binary test $X_i < u$, $u$ is a real number observed in the data
- Leaf nodes are associated with univariate distributions of the child
An example regression tree for a Module network

Module 1

MSFT

Module 2

MOT

AMAT

INTL

Module 3

DELL

HPQ

\[ P(M_3 | AMAT, INTL) \]

- **AMAT < 5%**
  - False
  - True

- **INTL < 4%**
  - False
  - True

\[ N(1.4, 0.8) \quad N(0, 1.6) \quad N(-2, 0.7) \]

Module 3 values are modeled using Gaussians at each leaf node.
A very simple regression tree

\[
\begin{align*}
X_2 &> e_1 \\
X_2 &> e_2
\end{align*}
\]

\[
\begin{align*}
\mu_1, \sigma_1 & \\
\mu_2, \sigma_2 & \\
\mu_3, \sigma_3 &
\end{align*}
\]
A regression tree to capture a CPD $P(X_3|X_1, X_2)$

$e_1, e_2$ are values seen in the data

Expression of gene represented by $X_3$ modeled using Gaussians at each leaf node
Algorithm for growing a regression tree

- Input: dataset $D$, child variable $X_i$, candidate parents $C_i$ of $X_i$
- Output: Tree $T$
- Initialize $T$ to a leaf node, $\mu, \sigma$ estimated from all samples of $X_i$
- While not converged
  - For every leaf node $l$ in $T$
    - Find $X_j \in C_i$ with the best split at $l$
    - If split improves score
      - add two leaf nodes, $i$ and $j$ below $l$
      - Update samples and parameters associated with $i$, $i$ and $j$
Learning a regression tree

• Assume we are searching for the parents of a variable $X_3$ and it already has two parents $X_1$ and $X_2$
• $X_4$ will be considered using “split” operations of existing leaf nodes

$N_l$: Gaussian associated with leaf $l$
Convergence in regression tree depends on

- Depth of tree
- Improvement in score
- Maximum number of parents
- Minimum number of samples per 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 are recovered in learned module network

• Using 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
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 ≥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.
Goals for this lecture

- Per-module network inference methods
- Module networks
- A few case studies of Module networks
- Combining per-gene and per-module network inference methods
Application of Module networks to yeast expression data

Candidate regulators

Expression data

Clustering

Gene partition

Module network procedure

Pre-processing

Regulator selection

Data selection

Gene reassignment to modules

Functional modules

Motif search

Annotation analysis

Hypotheses & validation

Graphic presentation

Post-processing

Regulation program learning

Conditions
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
Application of Module networks to mammalian data

- Module networks have been applied to mammalian systems as well
- We will look at a case-study in the human blood cell lineage
- Dataset
  - Genome-wide expression levels in 38 hematopoietic cell types (211 samples)
  - 523 candidate regulators (Transcription factors)

Human hematopoietic lineage

Novershtern et al., Cell 2011
Expression profiles of 80 transcriptional modules
An HSCs, MEPs, and Early Erythroid-Induced Module

PBX1, SOX4 need to be high and MNDA need to be low for the highest expression of these genes.
Other key points from this analysis

• Many novel regulators associated with the hematopoietic lineage
• Several regulators were validated based on shRNA and ChIP-seq analysis
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
• Bayesian networks were among the first type of PGMs for representing networks
• Applying Bayesian networks to expression data required several additional considerations
  – Too few samples: Sparse candidates, Module networks
  – Too many parents: Sparse candidates
  – Imposing modularity: Module networks
  – Assessing statistical significance: bootstrap
References


