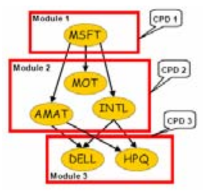
Computational Genomics

10-810/02-710, Spring 2009

Learning Module Networks

Eric Xing

Lecture 27, April 27, 2009



Reading: handouts

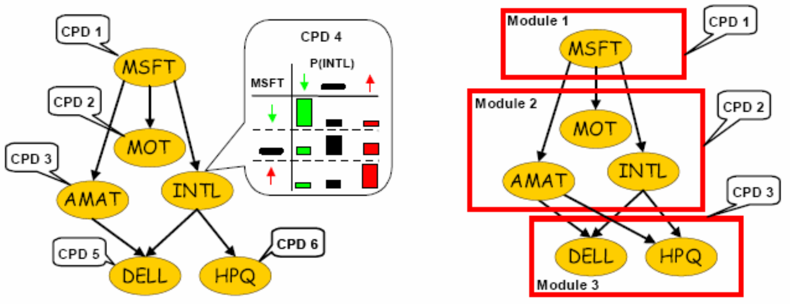
Gene Regulatory Network

- Aim : Identify Gene Regulatory Network of a cell from gene expression (microarray) data
- Classic Approach : Learn Bayesian Network
- Problem : Many genes, very little data points (eg. *S. cerevisiac* microarray data available for 2355 genes, and 173 arrays only)
- Bayesian Networks very noisy, edges not reliable
- Large Network usually very unstructured =>Edges are hard to interpret, and visualize.

Gene Regulatory Modules

- A cell's activity is organized as a network of interacting modules.
- Hence, identify modules of correlated genes, and build a network.
- Genes in same module share same parents and conditional probability distribution.
- Reduces hypothesis space of networks, and parameter space of CPTs, making learnt network more robust and reliable.

Example Of Module Network



(a) Bayesian network (b) Module network

1: (a) A simple Bayesian network over stock price variables; the stock price of Intel (*INTL*) is annotated with a visualization of its CPD, described as a different multinomial distribution for each value of its influencing stock price Microsoft (*MSFT*). (b) A simple module network; the boxes illustrate modules, where stock price variables share CPDs and parameters. Note that in a module network, variables in the same module have the same CPDs but may have different descendants.

THE MODULE NETWORK

A module network consists of :

- A Module Set *C*
  - *C* is the set of Module variables $M_1, M_2, \ldots, M_k$
- A Module Network Template *T* for *C*
  - A set of parents Pa_{Mj} for each module
  - A CPD for each module $P(M_j \mid Pa_{Mj})$
  - The network defined by *T* must be a DAG
- A Module Assignment Function *A* for *C*
  - An assignment of each variable X_i to a module M_j, i.e. $A(X_i) = M_j$

Learning the Module Network

- Training Data = *{x[1], x[2], …,x[M]}*
- *Aim :* Learn a module network structure *T* (composed of structure *S* and parameters *θ)* and assignment *A*
- Data Likelihood decomposed over modules :

$$L(\mathcal{M} : \mathcal{D}) = P(\mathcal{D} \mid \mathcal{M}) = \prod_{m=1}^{M} P(\mathbf{x}[m] \mid \mathcal{T}, \mathcal{A})$$

$$= \prod_{j=1}^{K} \left[\prod_{m=1}^{M} \prod_{X_i \in \mathbf{X}^j} P(x_i[m] \mid \mathbf{pa}_{\mathbf{M}_j}[m], \theta_{\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j}}) \right]$$

$$= \prod_{j=1}^{K} L_j(\mathbf{Pa}_{\mathbf{M}_j}, \mathbf{X}^j, \theta_{\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j}} : \mathcal{D}).$$

Bayesian Score

Model Score for a pair $(\mathcal{S}, \mathcal{A})$ defined as posterior probability, integrating out parameter *θ*

$$P(\mathcal{S}, \mathcal{A} \mid \mathcal{D}) \propto P(\mathcal{A}) P(\mathcal{S} \mid \mathcal{A}) P(\mathcal{D} \mid \mathcal{S}, \mathcal{A})$$

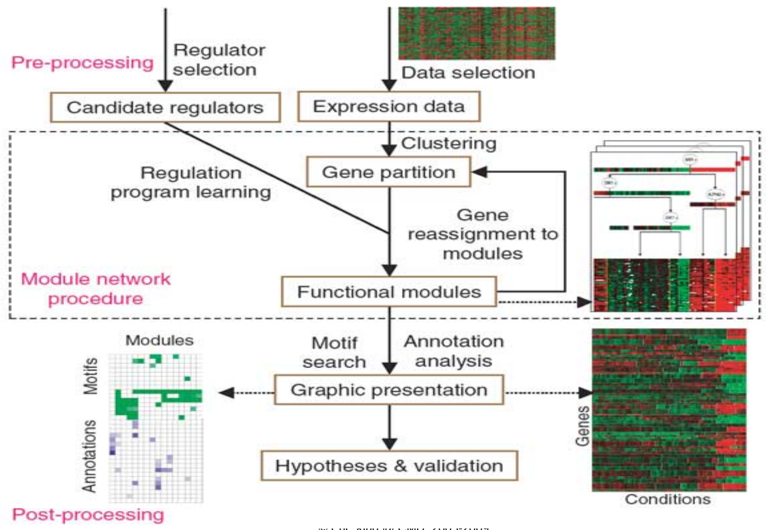
We define an assignment prior $P(\mathcal{A})$,
structure prior $P(\mathcal{S} \mid \mathcal{A})$ and a parameter prior $P(\theta \mid \mathcal{S}, \mathcal{A})$

$$P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}) = \int P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}, \theta) P(\theta \mid \mathcal{S}) d\theta.$$

Learning Algorithm

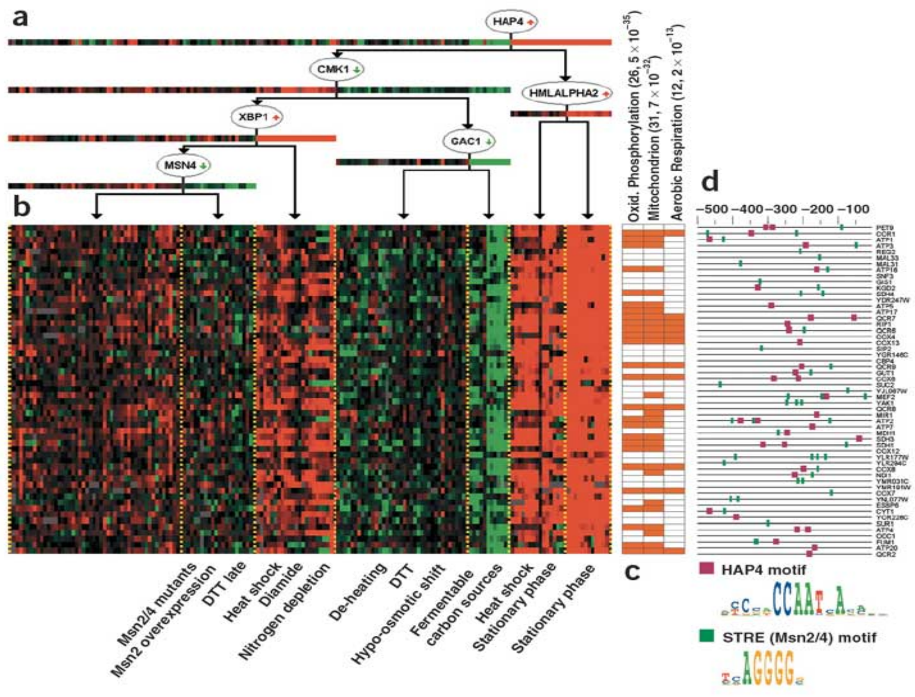
- Input : Data, *K* (no. of modules)
- Pick an initial assignment A_0 of nodes *X* to modules *M (by clustering similar features together)*
- Loop *t=1,2, …* till Convergence :
  - S_t = Pick best structure using $A_{(t-1)}$ and $S_{(t-1)}$
  - A_t = Pick best assignments using $A_{(t-1)}$ and S_t
- Return $M = (A_t, S_t)$

The Module Networks Algorithm.



Experiments

- Yeast dataset
- 2355 genes, 173 arrays under different stress conditions.
- Used 466 candidate regulators (found from Yeast Proteome Database)
- Regulator == Transcription Factor or Signaling Protein that may have transcriptional impact.
- Automatically infer 50 modules
- Used a Regression Tree, with Gaussian Distribution on Leaf Nodes, to represent $P(M_j \mid Pa_{Mj})$
- Experimental Validation done via (1) enrichment for cis-regulatory binding site motifs and (2) enrichment for GO annotations.



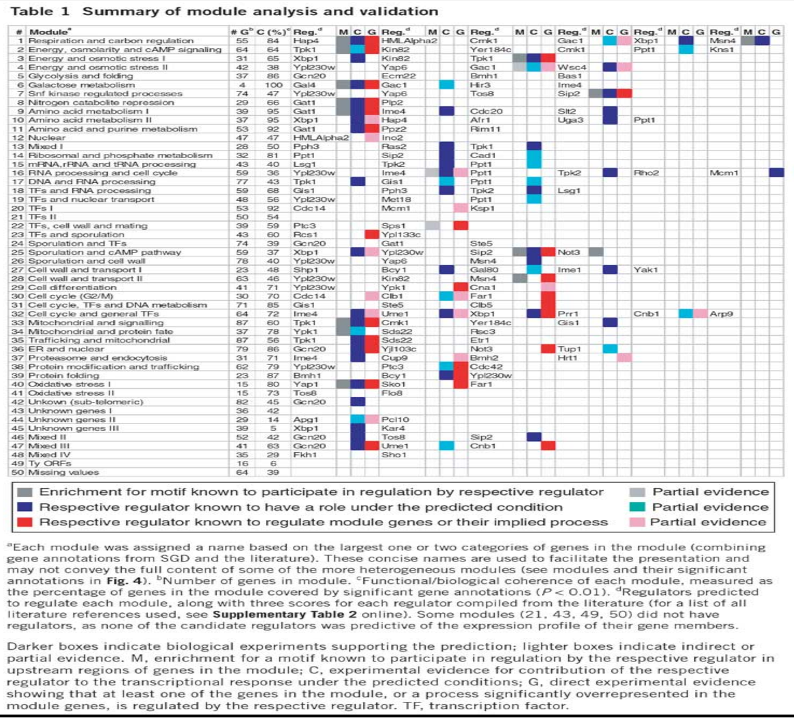


Table 1 Summary of module analysis and validation

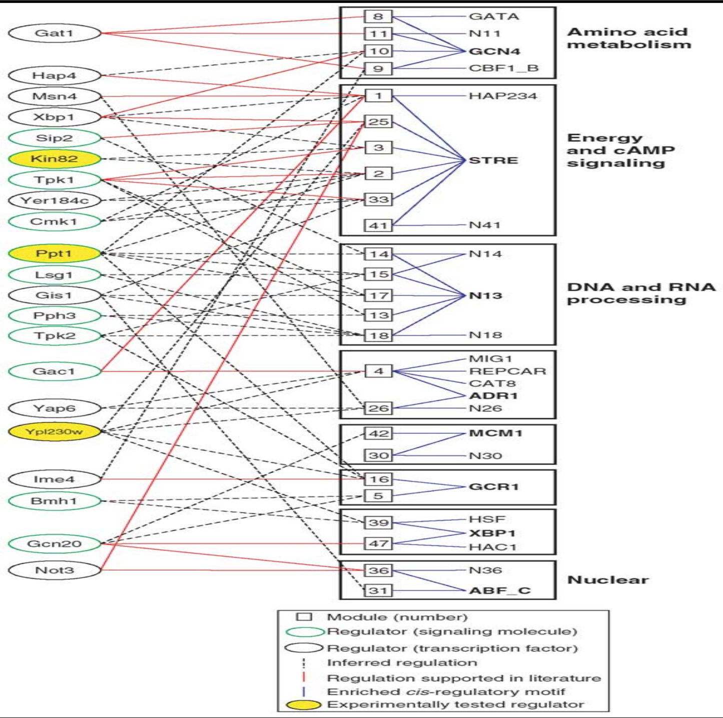
#	Module[a]	# G[b]	C (%)[c]	Reg.[d]	M	C	G	Reg.[d]	M	C	G	Reg.[d]	M	C	G	Reg.[d]	M	C	G	Reg.[d]	M	C	G	Reg.[d]	M	C	G
1	Respiration and carbon regulation	55	84	Hap4	⬛	🟦	🟥	HMLAlpha2				Cmk1				Gac1		🟩	🟪	Xbp1		🟦		Msn4	⬛	🟦	
2	Energy, osmolarity and cAMP signaling	64	64	Tpk1	⬛	🟩	🟥	Kin82				Yer184c				Cmk1				Ppt1		🟩		Kns1			
3	Energy and osmotic stress I	31	65	Xbp1		🟦		Kin82				Tpk1	⬛	🟦	🟥												
4	Energy and osmotic stress II	42	38	Ypl230w				Yap6				Gac1	▨	🟩	🟪	Wsc4		🟦	🟪								
5	Glycolysis and folding	37	86	Gcn20				Ecm22				Bmh1				Bas1											
6	Galactose metabolism	4	100	Gal4	⬛	🟦	🟥	Gac1		🟩		Hir3				Ime4											
7	Snf kinase regulated processes	74	47	Ypl230w				Yap6				Tos8				Sip2	⬛	🟦	🟥								
8	Nitrogen catabolite repression	29	66	Gat1	⬛	🟦	🟥	Plp2																			
9	Amino acid metabolism I	39	95	Gat1	⬛	🟦	🟥	Ime4		🟦		Cdc20				Slt2		🟦									
10	Amino acid metabolism II	37	95	Xbp1		🟦	🟪	Hap4				Afr1				Uga3		🟦		Ppt1							
11	Amino acid and purine metabolism	53	92	Gat1		🟦	🟥	Ppz2				Rim11															
12	Nuclear	47	47	HMLAlpha2			🟪	Ino2																			
13	Mixed I	28	50	Pph3				Ras2		🟦		Tpk1		🟦													
14	Ribosomal and phosphate metabolism	32	81	Ppt1				Sip2		🟦		Cad1		🟩													
15	mRNA,rRNA and tRNA processing	43	40	Lsg1				Tpk2		🟦		Ppt1		🟩													
16	RNA processing and cell cycle	59	36	Ypl230w				Ime4	▨	🟦	🟪	Ppt1				Tpk2		🟦		Rho2				Mcm1			🟦
17	DNA and RNA processing	77	43	Tpk1		🟦		Gis1		🟩		Ppt1		🟩													
18	TFs and RNA processing	59	68	Gis1				Pph3		🟦		Tpk2		🟦		Lsg1											
19	TFs and nuclear transport	48	56	Ypl230w				Met18				Ppt1		🟩													
20	TFs I	53	92	Cdc14				Mcm1			🟪	Ksp1															
21	TFs II	50	54																								
22	TFs, cell wall and mating	39	59	Ptc3				Sps1	▨		🟥																
23	TFs and sporulation	43	60	Rcs1			🟥	Ypl133c																			
24	Sporulation and TFs	74	39	Gcn20				Gat1				Ste5															
25	Sporulation and cAMP pathway	59	37	Xbp1		🟦	🟪	Ypl230w				Sip2	⬛	🟦	🟥	Not3	⬛										
26	Sporulation and cell wall	78	40	Ypl230w				Yap6				Msn4		🟦													
27	Cell wall and transport I	23	48	Shp1				Bcy1		🟦		Gal80		🟩		Ime1		🟦		Yak1							
28	Cell wall and transport II	63	46	Ypl230w				Kin82				Msn4	⬛		🟥												
29	Cell differentiation	41	71	Ypl230w				Ypk1			🟥	Cna1			🟪												
30	Cell cycle (G2/M)	30	70	Cdc14			🟪	Clb1		🟩	🟪	Far1			🟥												
31	Cell cycle, TFs and DNA metabolism	71	85	Gis1				Ste5				Clb5			🟥												
32	Cell cycle and general TFs	64	72	Ime4		🟦	🟪	Ume1		🟦	🟪	Xbp1		🟦	🟥	Prr1				Cnb1		🟩	🟪	Arp9			
33	Mitochondrial and signalling	87	60	Tpk1	⬛	🟦	🟥	Cmk1				Yer184c				Gis1		🟦									
34	Mitochondrial and protein fate	37	78	Ypk1	⬛	🟩		Sds22				Rsc3															
35	Trafficking and mitochondrial	87	56	Tpk1		🟦	🟥	Sds22				Etr1															
36	ER and nuclear	79	86	Gcn20		🟦	🟥	Yjl103c				Not3			🟥	Tup1		🟩									
37	Proteasome and endocytosis	31	71	Ime4		🟦		Cup9			🟪	Bmh2				Hrt1			🟪								
38	Protein modification and trafficking	62	79	Ypl230w				Ptc3		🟩	🟥	Cdc42															
39	Protein folding	23	87	Bmh1				Bcy1		🟦	🟥	Ypl230w															
40	Oxidative stress I	15	80	Yap1	⬛	🟦	🟥	Sko1			🟥	Far1															
41	Oxidative stress II	15	73	Tos8				Flo8																			
42	Unkown (sub-telomeric)	82	45	Gcn20		🟦																					
43	Unknown genes I	36	42																								
44	Unknown genes II	29	14	Apg1		🟩	🟪	Pcl10																			
45	Unknown genes III	39	5	Xbp1		🟦		Kar4																			
46	Mixed II	52	42	Gcn20		🟦		Tos8				Sip2		🟦													
47	Mixed III	41	63	Gcn20		🟦	🟥	Ume1		🟩		Cnb1			🟥												
48	Mixed IV	35	29	Fkh1				Sho1																			
49	Ty ORFs	16	6																								
50	Missing values	64	39																								

⬛	Enrichment for motif known to participate in regulation by respective regulator	▨	Partial evidence
🟦	Respective regulator known to have a role under the predicted condition	🟩	Partial evidence
🟥	Respective regulator known to regulate module genes or their implied process	🟪	Partial evidence

[a]Each module was assigned a name based on the largest one or two categories of genes in the module (combining gene annotations from SGD and the literature). These concise names are used to facilitate the presentation and may not convey the full content of some of the more heterogeneous modules (see modules and their significant annotations in **Fig. 4**). [b]Number of genes in module. [c]Functional/biological coherence of each module, measured as the percentage of genes in the module covered by significant gene annotations ($P < 0.01$). [d]Regulators predicted to regulate each module, along with three scores for each regulator compiled from the literature (for a list of all literature references used, see **Supplementary Table 2** online). Some modules (21, 43, 49, 50) did not have regulators, as none of the candidate regulators was predictive of the expression profile of their gene members.

Darker boxes indicate biological experiments supporting the prediction; lighter boxes indicate indirect or partial evidence. M, enrichment for a motif known to participate in regulation by the respective regulator in upstream regions of genes in the module; C, experimental evidence for contribution of the respective regulator to the transcriptional response under the predicted conditions; G, direct experimental evidence showing that at least one of the genes in the module, or a process significantly overrepresented in the module genes, is regulated by the respective regulator. TF, transcription factor.

12



So Far

- Proposed Notion of Module Networks
- Module N/ws restrict space of dependency structure, increase parameter sharing, and allow more robust models to be learnt.
- Modules may have biological meaning associated with them
- Limitations :
  1. Objective Function is not convex, learnt model is local maxima.
  2. Each Gene can be part of only 1 module !

Coregulated Overlapping Process Model (COPR)

- A single gene may participate in multiple modules, due to overlapping biological processes.
- Shifts from a Bayesian Network Perspective to a Probabilistic Relational Model (PRM)
- Models the assignment of genes to multiple overlapping processes, and the regulatory program associated with each process.
- The process is the equivalent of a module, except that a gene can belong to multiple processes.
- Makes the assumption that regulation is done at process level.

Gene Expression Model

- A set of *n* genes $\mathbf{G} = \{g_1, \ldots, g_n\}$
- A set of *k* array objects $\mathbf{A} = \{a_1, \ldots, a_k\}$
- A set of expression values of each gene, at each array :

$$\mathbf{E} = \{e_{1,1}, \ldots, e_{n,k}\}$$

- Allow a gene to participate in multiple processes, define

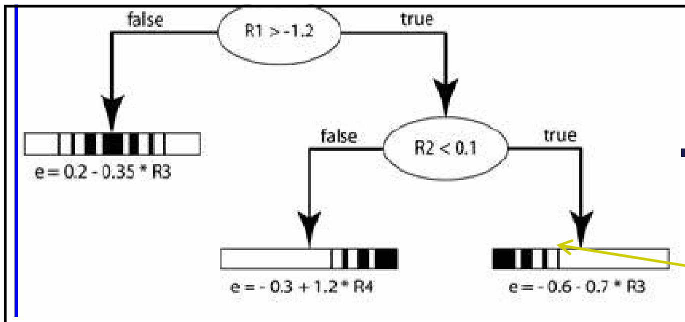
a set of binary *process membership* attributes $g.M_1, \ldots, g.M_j$

- Expression Level *e.level* for gene *g,* at array *a* is the sum of *g's* expression levels in all processes it participates, with Gaussian Noise added.

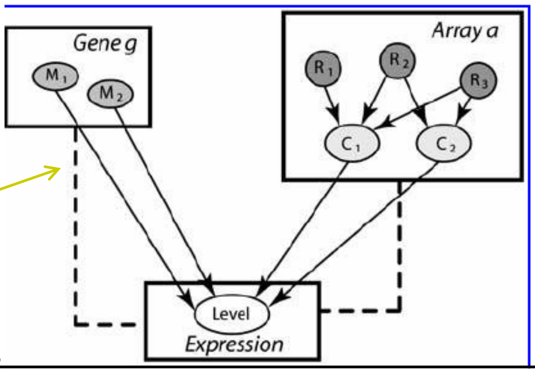
$$P(e.Level \mid g.\mathbf{M}, a.\mathbf{C}) = \mathcal{N}\left(\sum_{p=1}^{j} g.M_p \cdot a.C_p; \sigma_a^2\right)$$

Regulatory Model

- The activity level of process *p* in array *a* i.e. $a.C_p$ is a function of some regulatory program.
- Maintain *t* candidate regulators,having expression $a.R_r$ in array *a.*
- For each process *p*, we have *d* parent regulators, so that $a.C_p = f(R_{p,1}, R_{p,2}, \ldots, R_{p,d})$
- Like Module Networks, again use regression tree for the same.

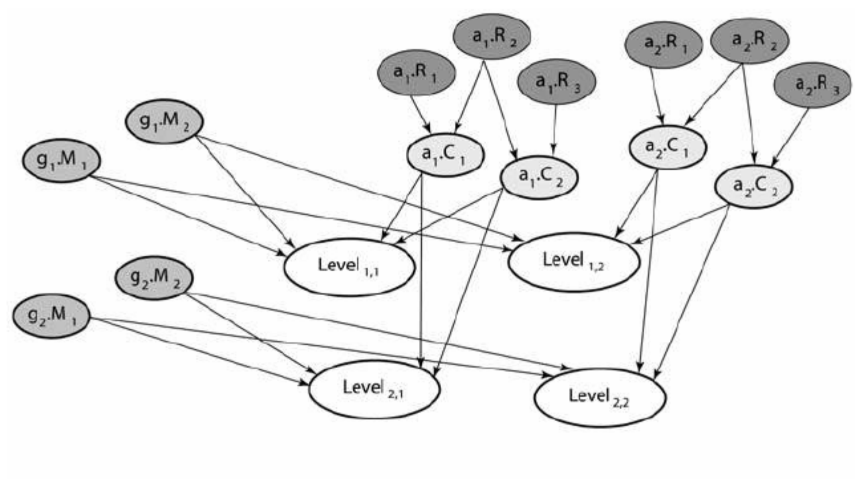


A Sample Regression Tree for a process with 2 regulators



A Sample COPR with 2 processes *(C_1, C_2)*, and 3 regulators *(R_1, R_2, R_3).* *M_1* and *M_2* denote whether gene *g* participates in processes 1 and 2 resp.

Instantiation of the Previous COPR for 2 Genes and 2 Arrays



COPR Model Summary

$$P(\mathbf{G}.\mathrm{M}, \mathbf{A}.\mathrm{C}, \mathbf{A}.\mathrm{R}, \mathbf{E}.Level) = \prod_{p=1}^{j} \prod_{g \in \mathbf{G}} P(g.M_p)$$

Prior on gene *g* in process *p*

Regulators Expression

$$\cdot \prod_{a \in \mathbf{A}} P(a.C_p \mid a.R_{i_1}, \ldots, a.R_{i_d}) P(a.R_{i_1}, \ldots, a.R_{i_d})$$

$$\cdot \prod_{e \in \mathbf{E}} P(e.Level \mid e.Gene.\mathbf{M}, e.Array.\mathbf{C}).$$

Activity of a Process given its regulators

Expression level of gene *g* in array *a,* given which processes it belongs to

Learning the COPR Model

- Expression levels for each gene, for each array known.
- Assignment of genes to processes unknown, activity levels $a.C_p$ unknown.
- Use structural EM (SEM) to learn the model
- In the E step, find a completion of the values to the hidden variables, given the model.
- In the M step, re-estimate the model structure and parameters, given the hidden variables (the current completion of the values).

OP Experiments

- The OP Model : no regulatory programs learnt for the processes.
- Yeast data : 173 yeast microarrays, 1010 genes, that had significant changes in gene expression.
- Learnt 30 processes.
- 24 genes predicted to be in no process, 552 in one process, 257 in two, and 119 in three, and 58 in four or more processes.
- Compare to Plaid Model (where a gene can belong to more than 1 cluster) and hierarchical clustering.

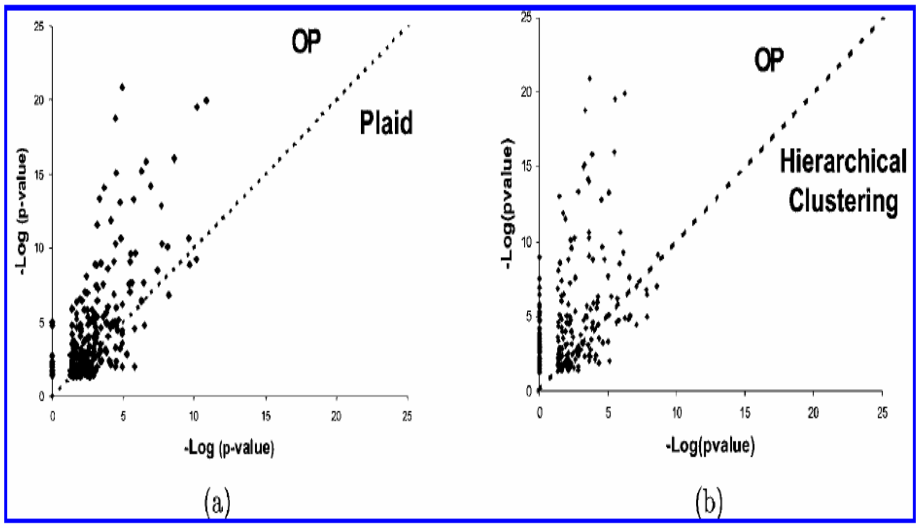


FIG. 3. Comparison of OP model to other approaches. (a) Scatter plot of the log p-value of different GO and KEGG nnotations for layers in Plaid on the one hand (X axis) and OP model processes on the other (Y axis). (b) Scatter lot of the log p-value of different GO and KEGG annotations for clusters from Pearson clustering on the one hand X axis) and OP model processes on the other (Y axis).

COPR Experiments

- The COPR Model : regulatory programs learnt for the processes.
- Yeast data : 173 yeast microarrays, 2034 genes.
- Learnt 50 processes.
- 1384 genes predicted to be in one process, 308 in two, 287 in three, and 40 in four or more processes.
- Compare to OP Model

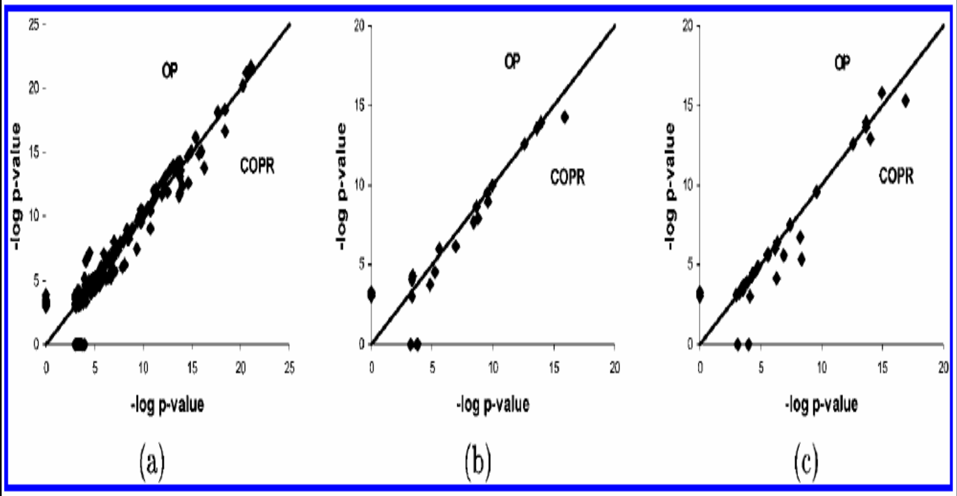


FIG. 5. Comparison of the simple OP and the COPR models on yeast stress data. Each graph shows a scatter plot of the negative log p-value for the enrichment of processes for different biological properties, comparing processes in the OP model (Y axis) and processes in the COPR model (X axis). (a) GO annotations; (b) presence of known motifs; (c) known transcription factor targets.

Comparison With Module Networks

- 394 yeast microarrays from 4 different studies
- Expression during cell cycle, various stress conditions, and in response to gene deletion mutations.
- COPR Model over 5747 genes, 50 processes, and 464 candidate regulators.
- COPR much better than Module Networks, since it allows genes to belong to multiple processes, that may be active under different experimental conditions.

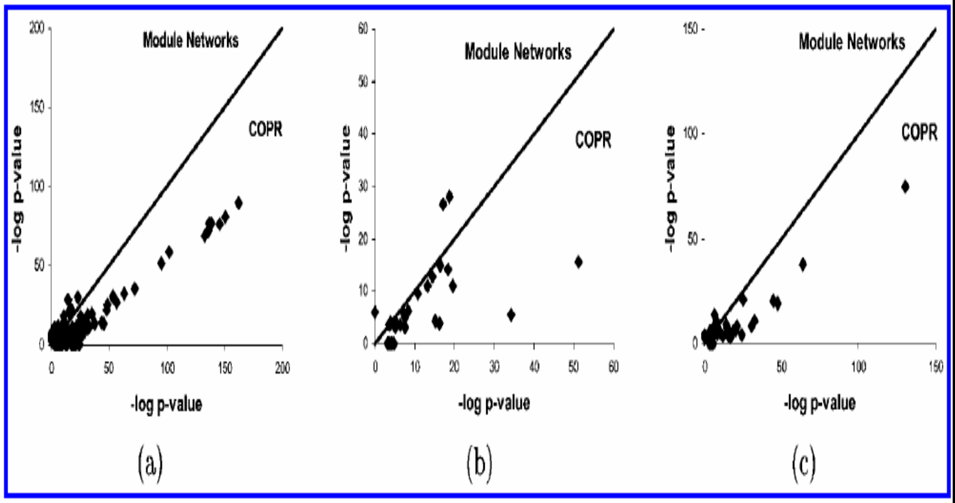
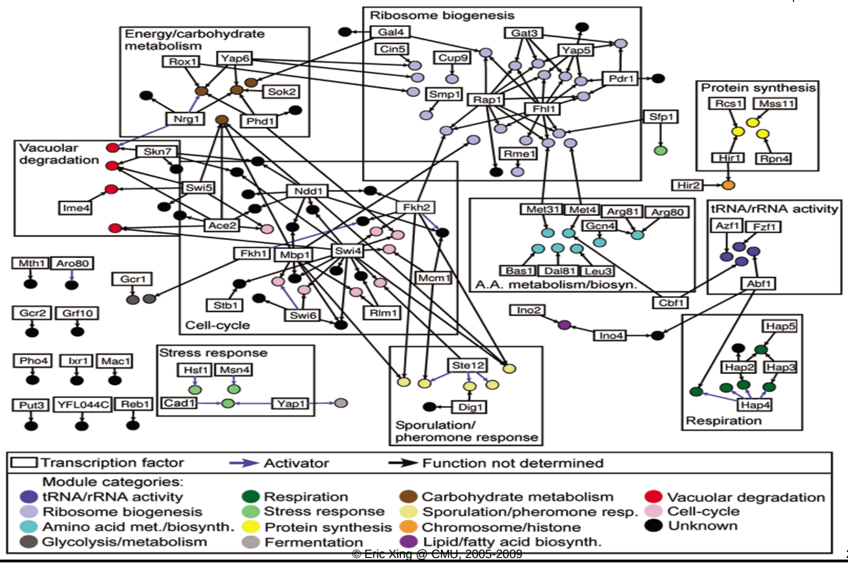
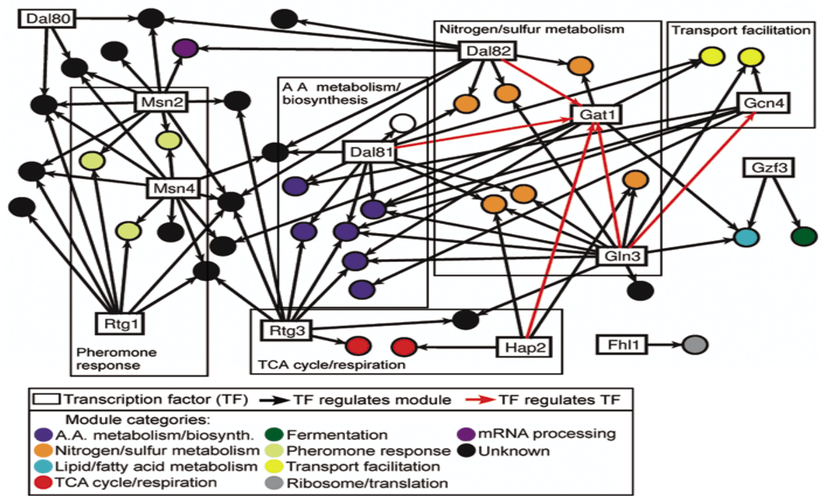


FIG. 6. Comparison of the module network model of Segal *et al.* to our COPR model on the yeast compendium data. Each graph shows a scatter plot of the negative log p-value for the enrichment of processes for different biological properties, comparing processes in the module network model (Y axis) and processes in the COPR model (X axis). (**a**) GO annotations; (**b**) presence of known motifs; (**c**) known transcription factor targets.

Computational discovery of gene modules and regulatory networks



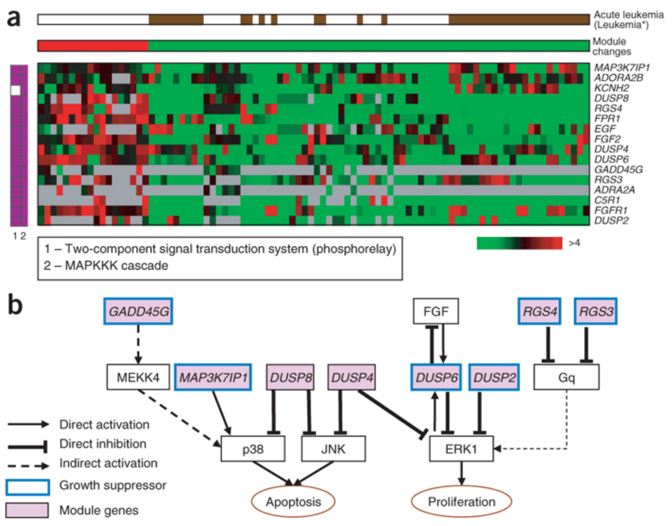
Analysis of the rapamycin transcriptional regulatory n/w



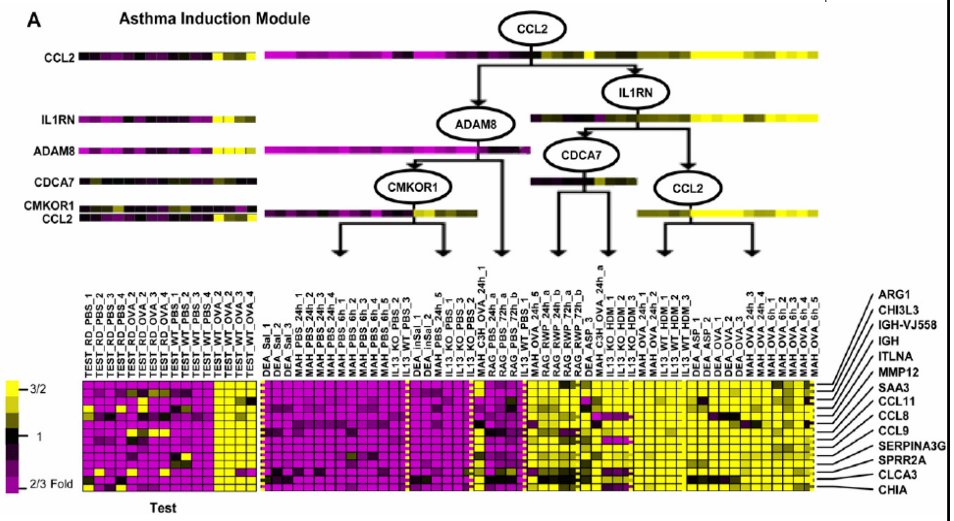
Conditional activity of expression modules in cancer

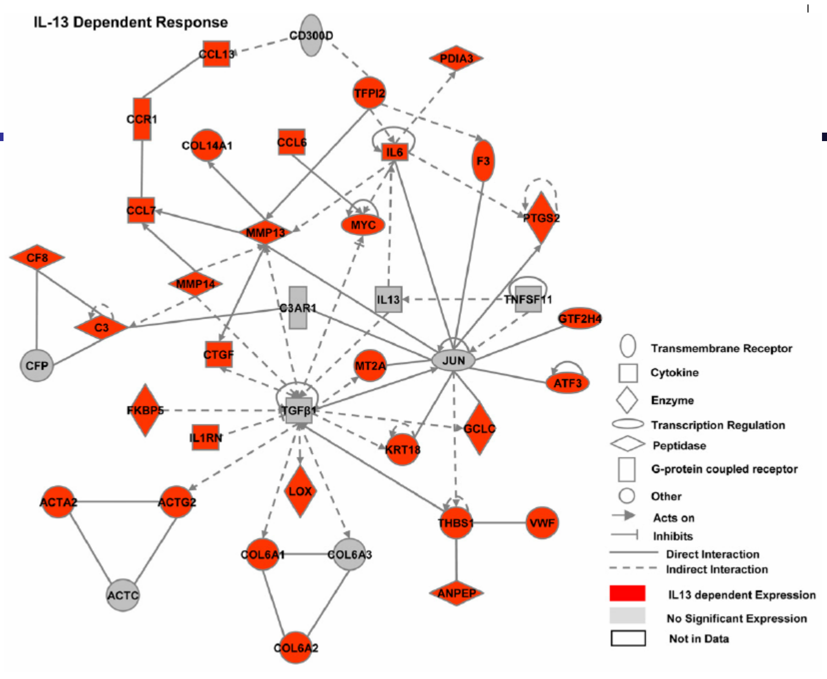
- Study 1975 microarrays across 22 tumor types, and 2849 genes.
- Activation of some modules is specific to particular types of tumor
- Eg: a growth-inhibitory module specifically repressed in acute lymphoblastic leukemias - may underlie the deregulated proliferation in these cancers.
- Other modules shared across a diverse set of clinical conditions, suggestive of common tumor progression mechanisms.
- Eg: the bone osteoblastic module

Growth Inhibitory Module



A Functional and Regulatory Map of Asthma





More Related Work

- Structure and evolution of transcriptional regulatory networks (Madan Babu et. al.) : studied the evolution of networks, via motifs and modules, and through extensive duplication of transcription factors and targets, with inheritance of regulatory interactions from the ancestral gene.
- The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets *de novo (Richard Bonneau et. al) :* derive genome-wide transcriptional regulatory interactions, via regression and variable selection to identify transcriptional influences on genes based on the integration of genome annotation and expression data

Conclusions

In this lecture, we looked at

- Representing sets of correlated genes together as a module, to form a module network.
- A module has the same set of parents, and the same CPD for all genes belonging to it.
- Learn a module via a greedy iterative algorithm.
- Looked at COPRs which allow a gene to belong to multiple modules instead of a single module.
- COPRs seem to be more capable of using multiple sources of data, measured under a variety of conditions.

References

- Learning Module Networks : Eran Segal, Dana Pe'er, Aviv Regev, Daphne Koller, Nir Friedman; JMLR, 6(Apr):557--588, 2005.
- Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data : Eran Segal, Michael Shapira, Aviv Regev, Dana Pe'er, David Botstein, Daphne Koller & Nir Friedman, Nature Genetics, 34, 166 - 176 (2003)
- Probabilistic Discovery of Overlapping Cellular Processes and Their Regulation, Alexis Battle, Eran Segal, Daphne Koller, Journal of Comp. Biology, Vol 12, 2005.
- Computational discovery of gene modules and regulatory networks, Ziv Bar-Joseph et. al., Nature Biotechnology 21, 1337 - 1342 (2003)
- A module map showing conditional activity of expression modules in cancer. Segal, Friedman, Koller, Regev, Nature Genetics, 2004
- A Functional and Regulatory Map of Asthma, Novershtern, Itzhaki, Manor, Friedman and Kaminski, American Journal of Respiratory Cell and Molecular Biology. Vol. 38, pp. 324-336, 2008

Additional Slides

Conditions for Score Decomposibility

- Globally Modular $P(\theta \mid \mathcal{S}, \mathcal{A}) = P(\theta \mid \mathcal{S})$

$$P(\mathcal{S}, \mathcal{A}) \propto \rho(\mathcal{S}) \kappa(\mathcal{A}) C(\mathcal{A}, \mathcal{S})$$

- Parameter Independence

$$P(\theta \mid \mathcal{S}) = \prod_{j=1}^{K} P(\theta_{\mathbf{M}_j | \mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S})$$

- Parameter Modularity

$$P(\theta_{\mathbf{M}_j | \mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}_1) = P(\theta_{\mathbf{M}_j | \mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}_2)$$

- Structure & Assignment Modularity

$$\rho(\mathcal{S}) = \prod_j \rho_j(\mathcal{S}_j), \qquad \kappa(\mathcal{A}) = \prod_j \kappa_j(\mathcal{A}_j)$$

Learning Modular N/ws : Details

- **Pick Best Structure :**
  - Start with previous structure $S_{(t-1)}$
  - Try using local operators (add an edge, delete an edge) to improve score
  - Stop when no local operator can improve score.
- **Pick Best Assignment :**
  - At each step, try to change the assignment of a single node i.e. change $A(X_i)$ to j from k.
  - If module network becomes cyclic, ignore the change.
  - If score improves, accept the change, else reject
  - Stop when improvement in score not possible.

Learning COPR Model : E-Step

- Find most likely joint assignment to *g.M*, and *a.C*
- Local Search Algorithm :
  - Fix *g.M*, find most likely *a.C*

$$\mathrm{argmax}_{\mathbf{G.M}} P(\mathbf{G.M} \mid \mathcal{M}) P(\mathbf{E}.Level \mid \mathbf{G.M}, \mathbf{A.C}, \mathcal{M})$$

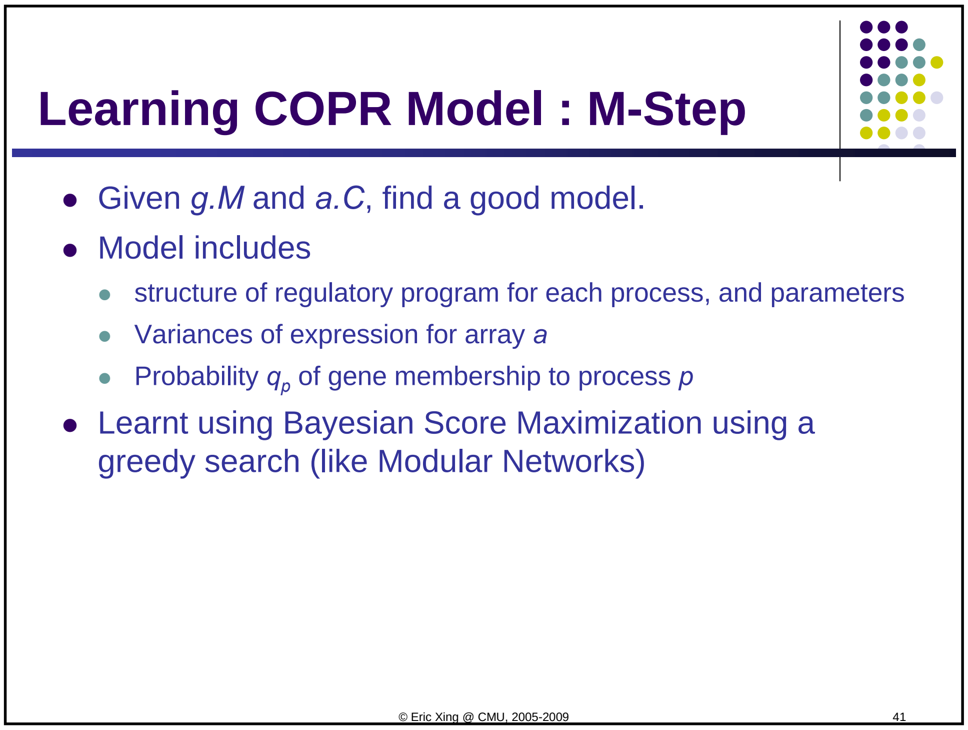
Decomposes so that we can maximize over each gene independently (still exponential in no. of processes, and requires linear relaxation to solve).

  - Fix *a.C*, find most likely *g.M*

$$\mathrm{argmax}_{\mathbf{A.C}} P(\mathbf{A.C} \mid \mathbf{A.R}, \mathcal{M}) P(\mathbf{E}.Level \mid \mathbf{G.M}, \mathbf{A.C}, \mathcal{M})$$

Reduces to minimize Least Squares, easy to optimize.

- Repeat till Convergence



Learning COPR Model : M-Step

- Given *g.M* and *a.C*, find a good model.
- Model includes
  - structure of regulatory program for each process, and parameters
  - Variances of expression for array *a*
  - Probability q_p of gene membership to process *p*
- Learnt using Bayesian Score Maximization using a greedy search (like Modular Networks)