Module networks ensembles for reverse engineering transcription regulatory networks

Tom Michoel, Anagha Joshi, Steven Maere, Eric Bonnet and Yves Van de Peer

Bioinformatics & Evolutionary Genomics
Department of Plant Systems Biology, VIB, Technologiepark 927, B-9052 Gent, Belgium
Department of Molecular Genetics, UGent, Technologiepark 927, B-9052 Gent, Belgium

Summary

- Module networks are probabilistic graphical models to infer transcription regulatory networks from gene expression data (Segal E, *et al.*: *Nat Genet* 2003, **34**:166):
  - genes are grouped in modules = groups of genes sharing the same parents and conditional distributions in the model.
  - regulators are arranged in regulation trees = conditional distributions are decision trees, testing up/down regulation of a parent at each node, with Gaussian distributions at the leaves.
- We have introduced a new learning method which decouples module discovery from regulation tree learning (Michoel T, *et al.*: *BMC Bioinformatics* 2007 **8** (Suppl 2): S5).
- Validation of heuristic optimization algorithms on synthetic data generated by SynTReN (Van den Bulcke T, *et al.*: *BMC Bioinformatics* 2006, **7**:43) shows high FP rate for all methods (Michoel T, *et al.*: *BMC Bioinformatics* 2007 **8** (Suppl 2): S5).
- We have extended our method and introduced a new probabilistic model which allows efficient sampling of module networks (Michoel T, *et al.*: *in preparation*).
- Statistical analysis of overrepresented edges in the network ensemble significantly improves TP rate of inferred regulatory interactions (Michoel T, *et al.*: *in preparation*).

Software

- LeMoNe v1: Java package for learning module networks using deterministic optimization and bottom-up learning of regulation trees (released under GPL).
- LeMoNe v2: Java package for learning module networks ensembles using Gibbs sampling and fuzzy decision trees (to be released).
- GaneSh: Java package for model based coclustering of genes and conditions using Gibbs sampling (released under GPL).

All software can be downloaded from our website:
http://bioinformatics.psb.ugent.be/software

Probabilistic model

$$p(x_1,\ldots,x_N \mid \{y_t\}) = \prod_k \prod_{i\in\mathcal{G}_k} p(x_i \mid \{y_t, t\in\mathcal{T}_k\}) \prod_{t\in\mathcal{T}_k} p(y_t \mid x_{r_t}, z_t, \beta_t),$$

- $p(x_1,\ldots,x_N \mid \{y_t\})$: expression levels
- $\prod_k$: genes in module k
- $\prod_{i\in\mathcal{G}_k}$: hidden variables (go left/right at decision tree node)
- $p(x_i \mid \{y_t, t\in\mathcal{T}_k\})$: Gaussian distribution conditioned on path in decision tree
- $\prod_{t\in\mathcal{T}_k} p(y_t \mid x_{r_t}, z_t, \beta_t)$: prob. to go left/right conditioned on up/down expression of regulator

- Introduction of hidden variables decouples module learning from parent learning.
- Module learning + EM on hidden variables = coclustering of genes and conditions.
- Sampling from posterior distribution = Gibbs sampling of coclusters + direct sampling of regulators for given coclusters.

Gibbs sampler for coclustering genes and conditions

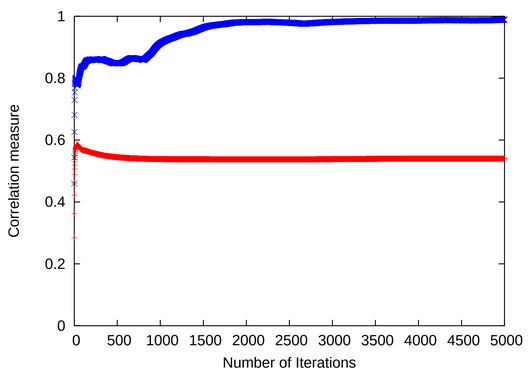

Figure 1: Correlation coefficient between two different Gibbs sampler runs for a small data set (top) and a large data set (bottom).

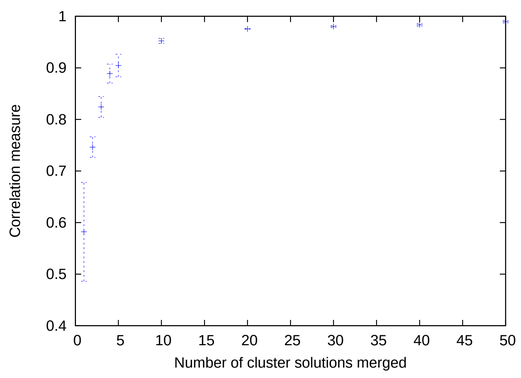

Figure 2: Correlation coefficient between different averages of the same number of local maxima for a data set with 1000 genes, 173 conditions.

- We use a Gibbs sampler which iteratively updates the assignment of genes to clusters, and within each gene cluster the assignment of conditions to condition clusters. The number of clusters is determined by the algorithm.
- For large data sets ($>$ 1000 genes, $>$ 100 conditions) the Gibbs sampler gets stuck in local maxima which partially overlap (Figure 1). The number of distinct local maxima is limited (Figure 2).
- The posterior distribution can be summarized as a set of fuzzy, overlapping clusters.

(Joshi A, *et al.*: *submitted*)

Sampling of regulators for given coclusters

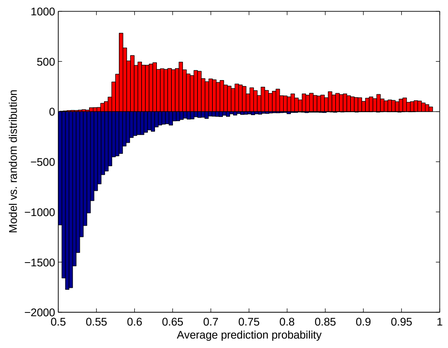

Figure 3: Histogram of the average probability that a regulator predicts the correct decision node direction, for samples from the model (red) and random samples (blue) (Gasch AP, *et al.*: *Mol Biol Cell* (2000), **11**: 4241) data set).

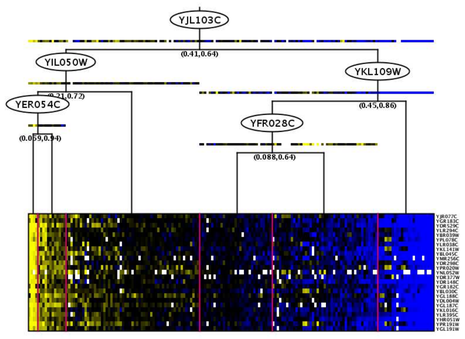

Figure 4: Regulation program learned from (Gasch AP, *et al.*: *Mol Biol Cell* (2000), **11**: 4241) data set.

- Given a set of coclusters and values of the hidden variables, regulators can be sampled from the posterior distribution for each decision tree node.
- Prediction probability of regulators can be used to prioritize high-scoring regulators, or suggest missing regulators where only low-scoring ones are found (Figure 3, 4).

Module networks ensemble analysis

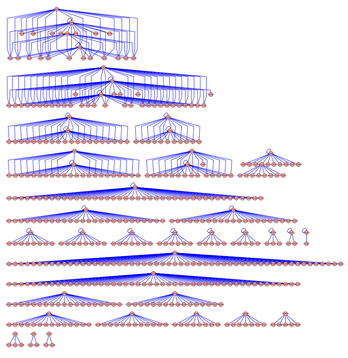

Figure 5: Highest weight network edges (SynTReN data)

- Construct weighted transcription regulatory network from the ensemble by drawing edges from each regulator to all the genes in that module; the weight is the prediction probability and weights for different module networks are added together.
- With high threshold on weight, network motifs (Shen-Orr SS, *et al.*: *Nat Genet* (2002), **31**:64) appear (Figure 5).
- Analysis of
  - synthetic data generated by SynTReN (Van den Bulcke T, *et al.*: *BMC Bioinformatics* 2006, **7**:43) (1000 nodes, 2361 edges),
  - public *E. coli* expression data compendium (Faith JJ, *et al.*: *PLoS Biology* (2007), **5**:0054) with RegulonDB (Salgado H, *et al.*: *Nucleic Acids Res* (2006), **34**: D394) as reference set of interactions,

  shows that interactions with higher weight have significantly higher TP rate (Figure 6).

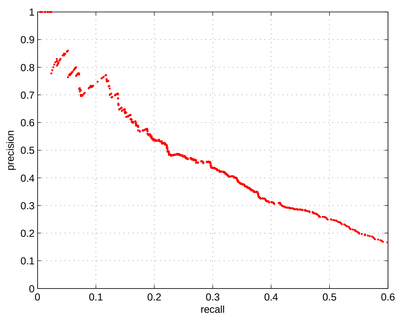

Figure 6: Recall – precision for different cutoffs on edge weights (SynTReN data)

(Michoel T, *et al.*: *in preparation*)