Module Networks, SynTren, etc.

Gent*[1], Leuven[2] and Antwerpen[3]

[1]Bioinformatics & Evolutionary Genomics, Department of Plant Systems Biology, VIB/Ghent University, Technologiepark 927, B-9052 Ghent, Belgium
[2]Leuven
[3]Antwerpen

Email: Gent*- ...@psb.ugent.be; Leuven - ...@kuleuven.ac.be; Antwerpen - ...@....be;

*Corresponding author

Abstract

Background:

Results:

Conclusions: c

Background

Results and Discussion

Overview

Synthetic data

Biological data

Conclusions

Methods

Data sets

SynTren [1] ...

Module networks

Module networks [2, 3] are a special kind of acyclic Bayesian networks in which groups of nodes, called *modules*, share the same parents and conditional distributions. For continuous gene expression values, the conditional distributions are normal with mean and standard deviation depending on the parent values through a regression tree that is called the *regulation program* of the module. The tests on the internal nodes of the regression tree are of the form $x \gtrless s$ for some split value s, where x is the expression value of the parent associated to the node.

The Bayesian score that is used to search for an optimal module network given the data is obtained by taking the log-marginal probability of the data likelihood over the parameters of the normal distributions at the leaves of the regression trees with a normal-gamma prior. The result [3] decomposes as a sum of leaf scores of the different modules:

$$S = \sum_k S_k$$
$$S_k = \sum_\ell \log \iint d\mu d\tau \, p(\mu,\tau) \prod_{m \to \ell} \prod_{i \in \mathcal{A}_k} p_{\mu,\tau}(x_{i,m}), \quad (1)$$

where k runs over the set of modules and ℓ runs over the leaves of the regression tree of module k; $p(\mu,\tau)$ is a normal-gamma distribution over the mean μ and precision τ of the normal distribution $p_{\mu,\tau}$, $m \to \ell$

denotes the experiments that end up at leaf ℓ after traversing the regression tree, and $i \in \mathcal{A}_k$ denotes the genes i assigned to module k.

An optimal module network is found heuristically by starting from an initial clustering, and alternatingly learning regression trees and reassigning genes to modules, iterating until convergence of the Bayesian score.

Learning regression trees

In [2, 3], regression trees are learned top-down by considering all possible splits on all current leaves with all possible candidate regulators that preserve acyclicity of the network. Here we learn regression trees for each module separately by a bottom-up hierarchical clustering of the experiments. The hierarchical tree gives a nested set of partitions from which the one with the highest score can be obtained. This defines the locations where the tree has to be cut. In a separate step, regulators are assigned to the internal nodes of the remaining tree.

At each step of the hierarchical clustering we have a collection of binary trees $\{T_\alpha\}_\alpha$ whose leaves consist of single experiments. By $\mathcal{E}_\alpha \subset \{1, \ldots, M\}$ we denote the subset of experiments forming the leaves of T_α. In considering a merge of two trees T_{α_1} and T_{α_2}, we are interested in the possible score gain

$$S_k(\mathcal{E}_{\alpha_1} \cup \mathcal{E}_{\alpha_2}) - S_k(\mathcal{E}_{\alpha_1}) - S_k(\mathcal{E}_{\alpha_2}), \qquad (2)$$

but to create a well balanced tree, we want to take into account the size of the trees T_{α_1} and T_{α_2}, and penalize the addition of small trees to already large trees.

For a trivial tree T_m consisting of a single leaf, $\mathcal{E}_m = \{m\}$, we set

$$Z_m = e^{S_k(\{m\})},$$

and define recursively for a binary tree T_α with child trees T_{α_1} and T_{α_2}, the quantity

$$Z_\alpha = e^{S_k(\mathcal{E}_\alpha)} + Z_{\alpha_1} Z_{\alpha_2}.$$

Note that

$$Z_\alpha = \sum_{\mathcal{P} \sim T_\alpha} e^{S_k(\mathcal{P})}, \qquad (3)$$

where $\mathcal{P} \sim T_\alpha$ denotes the partitions generated by the hierarchical subtree T_α and $S_k(\mathcal{P})$ is the score of the partition, defined as in eq. (1) with ℓ summing over the different sets in $\mathcal{P}$.

Then we define the merge score as

$$r_\alpha = \frac{e^{S_k(\mathcal{E}_\alpha)}}{Z_\alpha} = \frac{1}{1 + e^{-[S_k(\mathcal{E}_\alpha) - \ln Z_{\alpha_1} - \ln Z_{\alpha_2}]}}, \qquad (4)$$

and at each stage of the merging process merge those trees T_{α_1} and T_{α_2} with the highest merge score, i.e., those trees for which

$$S_k(\mathcal{E}_{\alpha_1} \cup \mathcal{E}_{\alpha_2}) - \ln Z_{\alpha_1} - \ln Z_{\alpha_2}$$

is largest.

As trees grow bigger, Z_α will contain more and more terms and therefore for a given score gain the highest merge score will be between the smallest trees, thus creating a well balanced tree. At the start there are M trees and $M(M-1)/2$ possible merges. In each subsequent step, the number of trees diminishes by 1, and in the list of possible merges, only those between existing trees and the newly formed tree have to be computed afresh.

To find recursively the highest scoring partition among all partitions generated by the hierarchical tree, note that for a given subtree T_α, the highest scoring partition is either the trivial partition into one set, or it is the union of the highest scoring partitions generated by the subtrees T_{α_1} and T_{α_2}. Hence the optimal partition can be constructed explicitly from the bottom up. Alternatively, to avoid overfitting the data by deep regression trees, we can start at the root and keep only those splits which significantly improve the score, i.e., keep a subtree split T_α if

$$S_k(\mathcal{E}_{\alpha_1}) + S_k(\mathcal{E}_{\alpha_2}) - S_k(\mathcal{E}_\alpha) > \epsilon,$$

for some cut-off $\epsilon \geq 0$.

Our method of hierarchical clustering is quite similar to the Bayesian hierarchical clustering of [4], except that in their method, partitions generated by a subtree get a different weight in the form of a prior probability, with partitions consisting of many smaller sets getting a lower weight. We have found that for our problem, giving equal weight to all partitions gives a slightly higher final score.

Regulator assignment

A given internal node T_α partitions its experiment set $\mathcal{E}_\alpha$ into two distinct sets $\mathcal{E}_{\alpha_1}$ and $\mathcal{E}_{\alpha_2}$ according

to the tree structure. Given a regulator r and split value s, we can also partition $\mathcal{E}_\alpha$ into two sets

$$\mathcal{R}_1 = \{m \in \mathcal{E}_\alpha \colon x_{r,m} \leq s\}$$
$$\mathcal{R}_2 = \{m \in \mathcal{E}_\alpha \colon x_{r,m} > s\},$$

where $x_{r,m}$ is the expression value of regulator r in experiment m.

Consider now two random variables: E which can take the values α_1 or α_2, and R which can take the values 1 or 2, with probabilities defined by simple counting, $p(E = 1) = |\mathcal{E}_{\alpha_1}|/|\mathcal{E}_\alpha|$, $p(R = 1) = |\mathcal{R}_1|/|\mathcal{E}_\alpha|$, etc. We are interested in the uncertainty in E given knowledge (through the data) of R, i.e., the conditional entropy [5]

$$H(E \mid R) = \frac{|\mathcal{R}_1|}{|\mathcal{E}_\alpha|} h(p_1) + \frac{|\mathcal{R}_2|}{|\mathcal{E}_\alpha|} h(p_2), \qquad (5)$$

where h is the entropy function

$$h(p) = -p \log(p) - (1-p) \log(1-p),\ 0 \leq p \leq 1,$$

and p_i are the conditional probabilities

$$p_i = p(E = \alpha_1 \mid R = i) = \frac{|\mathcal{E}_{\alpha_1} \cap \mathcal{R}_i|}{|\mathcal{R}_i|},\ i = 1, 2.$$

The conditional entropy is nonnegative and reaches its minimum 0 when $p_1 = 0$ or 1 (and consequently $p_2 = 1$, resp. 0), which means the $\mathcal{E}$ and $\mathcal{R}$ partitions are equal and the regulator – split value pair 'explains' the split in the regression tree exactly.

Hence to each intenal node of a regression tree we assign the regulator – split value pair which minimizes the conditional entropy (5) with the constraint that the module network remains acyclic. Since this assignment has to be done only once, after the module networks score has converged, the best regulator – split value pairs can be found by simply enumerating over all possibilities, even for relatively large data sets.

Because of the acyclicity constraint, the order in which regulators are assigned to nodes can influence the final result. Therefore we start by finding the optimal regulators for the roots of all trees, and assign regulators in the order of increasing entropy, the lowest first. Whenever the assignment of the best regulator to a root causes a cycle because of the previous assignments, we find a new best one that preserves acyclicity. After all the roots have got a regulator, the procedure is repeated for all the children of the roots, then for the children of the children, etc. We work our way down the trees like this because nodes higher in the trees will in general have a higher minimal entropy (as the partitioning is over more experiments), yet could be biologically more significant (because they are responsible for the most significant split between experiments), and should therefore be assigned their favorite regulator with higher probability.

Software

The latest version of SynTReN can be downloaded from [6] and the latest version of Genomica from [7].

Authors contributions

Acknowledgements

References

1. Van den Bulcke T, Van Leemput K, Naudts B, van Remortel P, Ma H, Verschoren A, De Moor B, Marchal K: **SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms**. *BMC Bioinformatics* 2006, **7**:43.
2. Segal E, Shapira M, Regev A, Pe'er D, Botstein D, Koller D, Friedman N: **Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data**. *Nat. Genet.* 2003, **34**(2):166 – 167.
3. Segal E, Pe'er D, Regev A, Koller D, Friedman N: **Learning module networks**. *Journal of Machine Learning Research* 2005, **6**:557 – 588.
4. Heller K, Ghahramani Z: **Bayesian hierarchical clustering**. In *Proceedings of the twenty-second International Conference on Machine Learning* 2005.
5. Shannon C: **A mathematical theory of communication**. *The Bell System Technical Journal* 1948, **27**:379 – 423, 623 – 656, [http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html].
6. **SynTReN**[http://homes.esat.kuleuven.be/~kmarchal/SynTReN/].
7. **Genomica**[http://genomica.weizmann.ac.il/].

Figures

Figure 1 - Sample figure title

A short description of the figure content should go here.

Figure 2 - Sample figure title

Figure legend text.

Tables

Table 1 - Sample table title

Here is an example of a *small* table in LaTeX using `\tabular{...}`. This is where the description of the table should go.

My Table		
A1	B2	C3
A2	...	..
A3	..	.

Table 2 - Sample table title

Large tables are attached as separate files but should still be described here.

Additional Files

Additional file 1 — Sample additional file title

Additional file descriptions text (including details of how to view the file, if it is in a non-standard format or the file extension). This might refer to a multi-page table or a figure.

Additional file 2 — Sample additional file title

Additional file descriptions text.