Finding Transcription Modules on Microarray Data Using PISA

Yangqiu Hu
yqhu@u.washington.edu
CSE 527 Final Project, 12/15/2004

Introduction

The technology of DNA microarray has made it possible to study the transcriptional response to different experimental conditions on a genome scale. A large amount of data has been accumulated, demanding powerful computational tools to extract the information and knowledge from it. While classical clustering methods (such as hierarchical clustering, k-means clustering, and self-organization map (SOM) have been successfully applied to finding genes that are co-regulated under a relatively small number of specific conditions, they are less effective when applied to large scale data mainly due to two limitations [5]. First, these clustering methods usually assign each gene to a single class, whereas genes may participate in more than one biological function and thus should be included in multiple clusters. Second, these methods measure the correlation in expression patterns over all condition, but genes are typically only regulated in specific experimental contexts, and the remaining conditions are simply the noise. Finding transcription modules from large-scale genomic data sets has fallen into one of the applications where common clustering methods may encounter inherent difficulties.

Many approaches have been devised to address these issues, including biclustering. Before this term was first used by Cheng and Church [9] in microarray data analysis, it had been studied in other fields under different names such as coclustering, bidimensinal clustering, and subspace clustering. Biclustering performs clustering on the genes and conditions simultaneously, which produces a local model as opposed to the global model generated by conventional clustering. Clearly biclustering is suitable for situations when (1) only a small set of genes participate in a cellular process; (2) a cellular process is active only in a subset of conditions; and (3) some genes may participate in multiple pathways that may or may not be activated under all conditions [3]. Therefore, biclustering may be a key technique for finding transcription modules.

Among the biclustering algorithms, the Signature Algorithm (SA) [5] and its variants, the Iterative Signature Algorithm (ISA) [3,4] and the Progressive Iterative Signature Algorithm (PISA) [6] are promising particularly due to their biological nature. In a transcription control network a single transcription factor (TF) typically regulates multiple genes, and a transcription module corresponds to a set of such genes and the conditions that are connected by the transcription factor. Thus the SA series algorithms are chosen as the main focus of this project, and its most advanced version, PISA, is implemented and tested on simulated and real data. The remainder of the report includes a brief overview of biclustering, followed by the details of SA/ISA/PISA and some

implementation issues, as well as the experimental results, and conclusions and future work.

A brief overview of biclustering

Bicluster is a subset of rows that exhibit similar behavior across a subset of columns, and vice versa. Given a data matrix, biclustering is the identification of a set of biclusters that meet some homogeneity criteria. See [1,2] for reviews of biclustering algorithms. The data matrix can be regarded as a weighted bipartite graph, where each row corresponds to a node $n_i \in$ L, and each column corresponds to a node $n_j \in$ R, where L and R are the bipartition of the graph. The only edge between n_i and n_j has weight a_{ij}, the entry of the data matrix. This formulation leads to the proof that finding the bicluster with maximum size is NP-complete, which is equivalent to finding the maximum edge biclique in a bipartite graph. For this reason, most biclustering algorithms employ heuristics to avoid the exponential time.

Biclusters can be of different types: (1) constant values; (2) constant values on rows or columns; (3) coherent values; (4) coherent evolutions. Biclusters can also have different structures: (1) exclusive rows and columns; (2) non-overlapping with checkerboard structure; (3) exclusive rows; (4) exclusive columns; (5) non-overlapping with tree structure; (6) non-overlapping nonexclusive; (7) overlapping with hierarchy; and (8) arbitrarily positioned overlapping.

Depending on the different heuristics approaches, biclustering can be further classified as (1) iterative row and column clustering; (2) divide and conquer; (3) greedy iterative search; (4) exhaustive enumeration; and (5) distribution parameter identification. Some common biclustering algorithms are publicly available, such as the Cheng and Church method [9], the Coupled Two-Way Clustering (CTWC) [10], the Plaid model [11], and the SAMBA [12].

The Signature Algorithm (SA)

The SA was introduced by Ihmels et al in 2002 [5]. Here the notion of a significant bicluster (or consistency) is intrinsically defined on the genes and the conditions – the conditions uniquely define the genes, and vice versa. A bicluster is formally defined as a module, which includes a set of co-regulated genes and a set of conditions that trigger this co-regulation. The SA receives as input a set of genes that partially overlap a TM, and it gives as output a complete set of module with gene signature and condition signature. There are two steps in the SA. Step 1 selects the conditions under which the input genes are most tightly co-regulated, which involves a scoring and a thresholding:

The condition score is computed using $s_c = \left\langle E_G^{gc} \right\rangle_{g \in G_I}, c = 1,...,C$.

And the thresholding on conditions is $S_C = \left\{ c \in C : \left| s_c - \left\langle s_c \right\rangle_{c \in C} \right| > t_C \sigma_C \right\}$.

Step 2 selects the genes whose expression level change significantly under the conditions selected in step 1. It also includes a scoring and a thresholding:

The gene score is computed using $s_g = \left\langle s_c E_C^{gc} \right\rangle_{c \in S_C}$.

And the thresholding on genes is $S_G = \left\{ g \in G : \left| s_g - \left\langle s_g \right\rangle_{g \in G} \right| > t_G \sigma_G \right\}$

These two steps will largely remove uncorrelated genes and conditions. However, the SA requires prior knowledge to compile the input gene set, and it may be difficult to determine the threshold values to use. In addition, it does no further iteration after the two steps. These drawbacks are overcome by the ISA.

The Iterative Signature Algorithm (ISA)

The ISA improves the SA in the following aspects [3,4]:

(1) Running SA iteratively;
(2) Starting each run of SA with random input sets; and
(3) Using a range of threshold values.

By doing these, the application of ISA can be fully automatic and requires no prior knowledge. Using a range of threshold values also provides the possibility of revealing the hierarchical modular organization of transcription control at different resolutions.

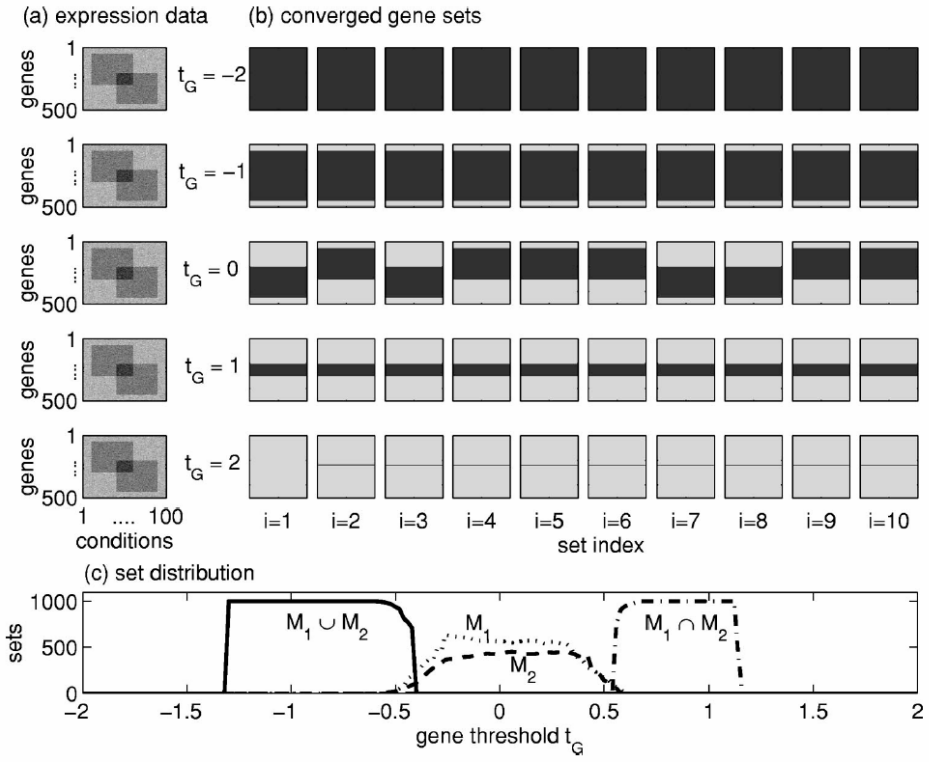

Figure 1. ISA's ability to revolve overlapping modules. From [4].

Figure 1 shows ISA's ability to resolve overlapping modules when used on simulated data. Ihmels et al also applied ISA to a *Saccharomyces Cerevisiae* microarray data set containing 6206 genes and 1011 experimental entries. They set t_G ranging between 1.8 and 4.0 at an interval of 0.1, and a fixed $t_C = 2.0$. They compiled ~20,000 random input gene sets, each generating a fixed point for each t_G. Finally a module fusion step is used to cluster the fixed points for each t_G, resulting in the final modules, as shown in figure 2.

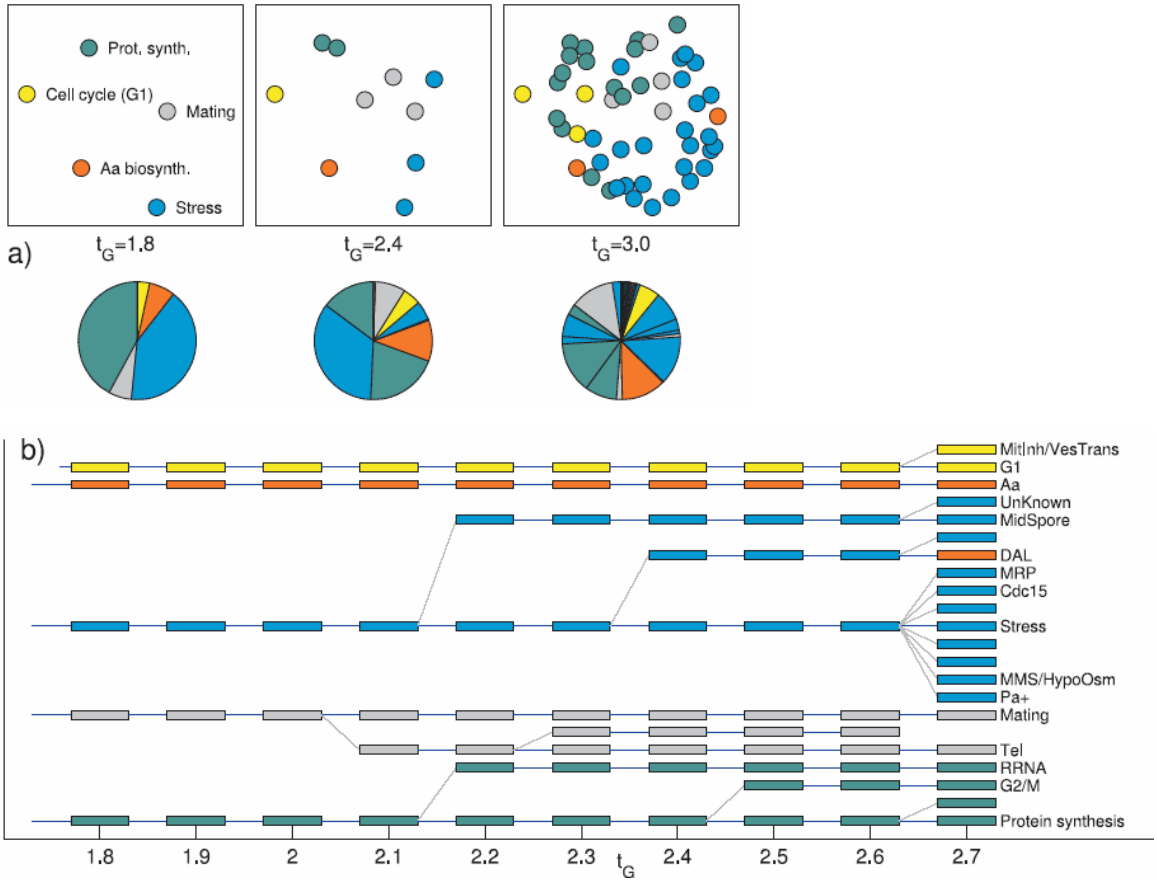

Figure 2. Hierarchical modular organization found by ISA. From [3].

The Progressive Iterative Signature Algorithm (PISA)

Despite its success, ISA still suffers from two major limitations: First, there are a lot of spurious fixed points. Second, weak modules might be overwhelmed by strong modules. To address these problems, Kloster et al developed the PISA [6]. Its central idea is that after finding a converged module, its contribution to the condition score should be removed before the next iteration. It is hoped that hidden or weak modules can be exposed after the orthogonalization. PISA also attempts to improve ISA by avoiding positive feedback, and enhancing the thresholding schema. There are four important steps in PISA:

1. Normalization

It is essential to make the expression data for each gene comparable, so the gene score threshold can be applied to all genes on an equal footing. For this purpose, E_G and E_C, two different copies of the raw expression matrix E, are produced for computing the gene and condition scores, respectively

$$\left(E'\right)_{gc} = \left(E\right)_{gc} - \left\langle \left(E\right)_{g'c} \right\rangle_{g'}$$

$$\left(E''\right)_{gc} = \left(E'\right)_{gc} - \left\langle \left(E'\right)_{g'c} \right\rangle_{g'}$$

$$\left(E_G\right)_{gc} = \left(E''\right)_{gc} \Big/ \sqrt{\left\langle \left(E''\right)^2_{gc'} \right\rangle_{c'}}$$

$$\left(E_{C,0}\right)_{gc} = \left(E_G\right)_{gc} \Big/ \sqrt{\left\langle \left(E_G\right)^2_{g'c} \right\rangle_{g'}}$$

2. PISAstep
PISAstep is a modified ISA. Compared to ISA, PISAstep only takes threshold on the gene scores, and since it uses a robust estimate of the mean and standard deviation, it eliminates the use of a range of threshold values. In addition, leave-one-out scoring is used to avoid positive feedback.

3. Orthogonalization
This is the major advantage of PISA over ISA. When a new module has been found, its contribution to the condition scores is removed by

$$E_C^{new} = E_C - E_C \frac{s^C \left(s^C\right)^T}{\left|s^C\right|^2}$$

4. Postprocessing
We use hierarchical clustering of the condition scores of the preliminary modules to generate consistent modules. One advantage of this approach is that we can determine the number of consistent modules.

The above algorithm of PISA is implemented in Matlab.

Experimental results

The implemented PISA is first used to process a simulated data set containing 5 overlapping modules (figure 3). PISA is able to identify 4 modules correctly, while does not find one module completely. This is probably due to the overlapping of both genes and conditions, and the addictive pattern of two linearly varying modules.

The PISA is then applied to the yeast data from Gasch et al [7]. The original data set has 6152 genes and 173 conditions. However, to compare the results of PISA with the 50 clusters defined in Segal et al [8], we only use the 2355 genes that they used. So the final data has 2355 genes and 173 conditions.

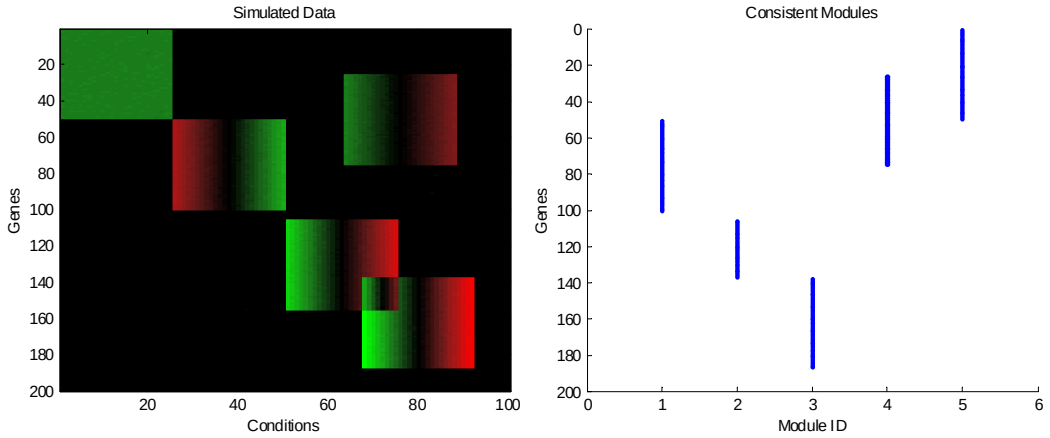

Figure 3. PISA applied to simulated data

We run the PISA 100 times and get 2210 preliminary modules. We then choose 50, 100, and 150 as the number of final modules, and use hierarchical clustering for the preliminary modules. Table 1 summaries the final results.

Table 1. PISA results

# modules	% genes included	max # overlapping mod.	mean module size
50	78.28%	11	99.76
100	89.20%	16	91.72
150	94.60%	24	95.31

To compare the results with the modules identified in [8], we compute the p-values for each category of the Gene Ontology database (ref.) that maximally overlaps with a module.

$$p = 1 - \sum_{i=0}^{n-1} \frac{\binom{c}{i}\binom{N_G - c}{m - i}}{\binom{N_G}{m}}$$

where Ng is the number of genes in organism (2355), m is the number of genes in module, c is the number of genes in GO category, and n is the number of genes in both module and GO category. We only compute GO categories with no more than 300 genes.

The results are plotted in figure 4. It is clear that PISA outperforms the modules found in [8], and as the number of modules increases, the biological relevance also increases. However, it is not clear how to determine the optimal number of modules.

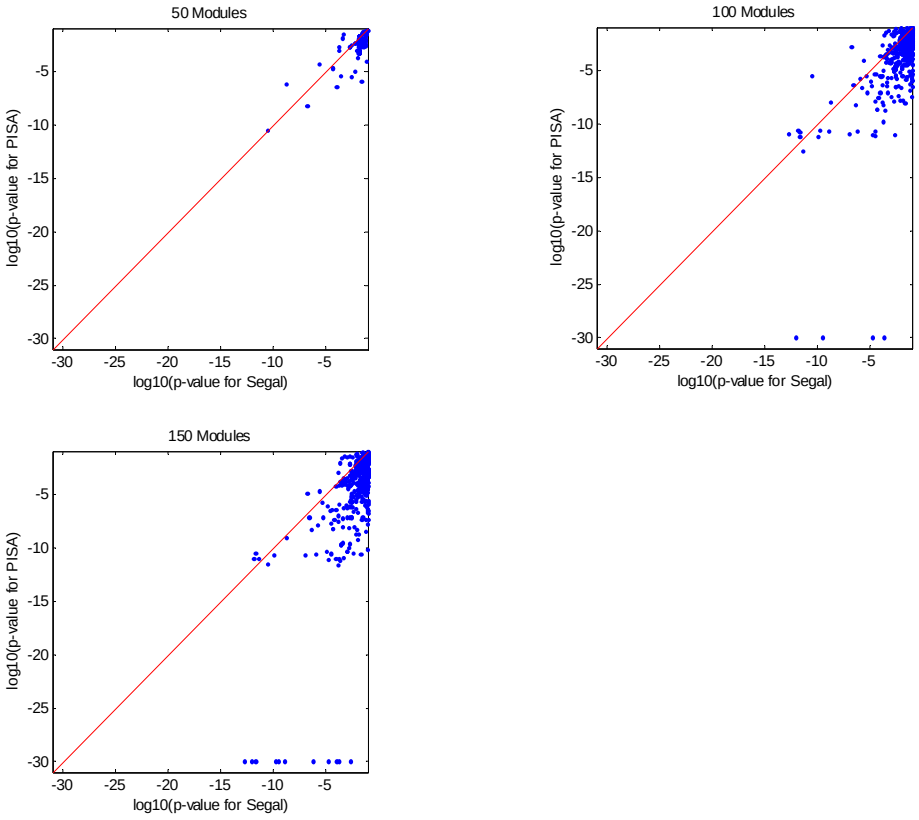

Figure 4. Comparing PISA's modules with Segal's modules

Conclusions

In this report I first discussed two limitations of classical clustering methods when applied to large scale genomic expression data, and then briefly reviewed biclustering that may be an alternative to clustering. Next I introduced in greater details the Signature Algorithm family (SA/ISA/PISA), among which I chose the most advanced version, PISA, to implement for the course project. My implementation was tested using simulated data, which partly confirmed its ability to resolve overlapping modules. Then the PISA was applied to the yeast data with 2355 genes and 173 conditions, and the results were compared with published results. Judging from the p-values of overlapping with GO annotations, the modules identified by PISA are more biologically relevant.

Future work may include determining the optimal number of modules, applying PISA to more data sets, the validation of biclustering methods, using both internal and external data, and comparing PISA with other biclustering methods.

References

[1] Tanay, A. et al, *Biclustering Algorithms: A Survey*, Handbook of Bioinformatics, 2004, to appear.

[2] Sara C. Madeira and Arlindo L. Oliveira, *Biclustering Algorithms for Biological Data Analysis: A Survey*, IEEE Transaction on Computational Biology and Bioinformatics, vol.1, no.1, 24-45, 2004
[3] Jan Ihmels, Sven Bergmann and Naama Barkai, *Defining transcription modules using large-scale gene expression data*, Bioinformatics 20(13):1993-2003 (2004)
[4] Sven Bergmann, Jan Ihmels and Naama Barkai, *Iterative signature algorithm for the analysis of large-scale gene expression data*, Phys. Rev. E 67, 031902 (2003)
[5] Jan Ihmels, Gilgi Friedlander, Sven Bergmann, Ofer Sarig, Yaniv Ziv and Naama Barkai, *Revealing Modular Organization in the Yeast Transcription Network, Nature Genetics* 31/4, 370-377 (2002)
[6] Kloster M, Tang C, Wingreen NS, *Finding regulatory modules through large-scale gene-expression data analysis*, Bioinformatics. 2004 Oct 28
[7] Gasch et. al., *Genomic expression programs in the response of yeast cells to environmental changes*, Mol Biol Cell. 2000 Dec;11(12):4241-57
[8] Segal et. al. *Module Networks: Identifying Regulatory Modules and their Condition Specific Regulators from Gene Expression Data*, Nat Genet. 2003 Jun;34(2):166-76
[9] Cheng and Church Biclustering: http://cheng.ececs.uc.edu/biclustering/
[10] CTWC: http://ctwc.weizmann.ac.il/
[11] Plaid model: http://www-stat.stanford.edu/~owen/plaid/
[12] SAMBA: http://www.cs.tau.ac.il/~rshamir/samba/