Applications of Spectral Matrix Theory in Computational Biology

Soheil Feizi

Computer Science and Artificial Intelligence Laboratory
Research Laboratory for Electronics
Massachusetts Institute of Technology

Abstract

In this report, we consider three applications of Spectral Matrix Theory in computational biology. First, we use spectral density functions of gene networks to infer their global structural properties. we observe eigenvalue distributions of these networks follow scale-free spectral densities with heavy positive tails. Second, we show how positive eigenvectors of modularity matrices can be used to highlight modules over networks. Finally, we use a spectral decomposition to deconvolve indirect information flows and infer direct interactions over networks.

I. Introduction

Using spectral matrix theory has become popular in analysis of biological networks since these methods usually leverage the entire structure of networks and usually are more robust. Here, we consider three applications of Spectral Matrix Theory in computational biology:

- In Section II, we use spectral density functions of gene networks to infer their global structural properties.
- In Section III, we use a spectral decomposition of modularity matrices to highlight modules over networks.
- In Section IV, we use a spectral decomposition of networks to deconvolve indirect information flows and infer direct interactions over them.

II. STRUCTURAL PROPERTIES OF GENE REGULATORY NETWORKS USING EIGENVALUE DISTRIBUTIONS

Understanding structural properties of biological networks is crucial to understand their functions. There have been a lot of prior work to analyze quantities such as degree distributions, clustering coefficients and shortest connecting paths. However, these quantities are loosely connected to global properties of these networks. Here, we use a similar approach to reference [2] to analyze structural properties of regulatory networks for human, fly and worm using their eigenvalue distributions.

The spectral density of a graph is the density of its eigenvalues defined as follows:

$$\rho(\lambda) = \frac{1}{N}\sum_{i=1}^{N}\delta(\lambda - \lambda_i), \tag{1}$$

where N is the number of nodes. As $N \to \infty$, $\rho(\lambda)$ converges to a continuous function. The spectral density of a graph is strongly connected to global structural properties of that graph. For example, suppose M_k is the k-th moment of $\rho(\lambda)$ defined as follows:

$$M_k = \frac{1}{N}\sum_{i=1}^{N}(\lambda_i)^k = \frac{1}{N}\mathrm{Tr}(A^k) = \frac{1}{N}\sum_{i_1,\ldots,i_k} A_{i_1,i_2}A_{i_2,i_3}\ldots A_{i_k,i_1}, \tag{2}$$

where A is the adjacency matrix of the graph. Therefore, NM_k is the number of directed loops over the graph (which is undirected).

Although the spectral density of an uncorrelated random graph converges to a semi-circle, spectral densities of biological networks do not follow the semi-circle law as illustrated in Figure 1-a. On the other hand, eigenvalue distributions of scale-free networks show a similar behavior (Figure 1-b), indicating the scale-free nature of gene regulatory networks.

Another interesting point can be observed by using odd moments of the spectral density function. Small odd moments indicates that the underlying undirected graph has fewer directed loops with odd lengths. As tree structures have zero directed loops with odd lengths, one can conclude that, networks with small odd spectral moments should have structural properties similar to trees.

Finally, note that these networks have larger positive eigenvalues compared to the negative ones. In the next section, we will discuss about the importance of these eigenvalues in forming modules over the network.

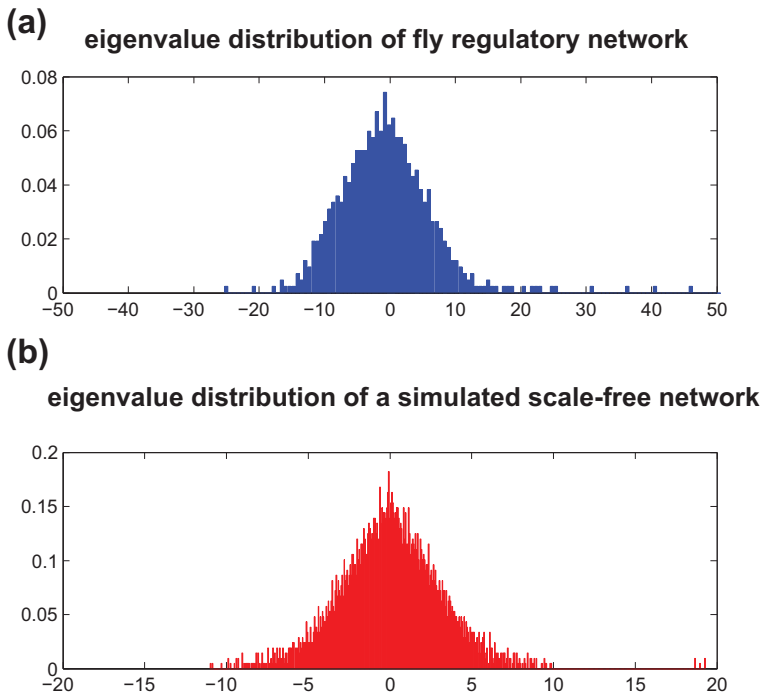

Fig. 1. Eigenvalue distribution of fly regulatory network
Panel (a) shows the spectral distribution of fly regulatory network. Panel (b) shows the eigenvalue distribution of a random scale-free graph constructed using prefrential attachment.

III. Network modularity using a spectral method

In this section, we review *spectral network modularity* algorithm which is based on some metrics defined in reference [1]. This algorithm assigns higher weights to interactions that are more likely to belong to modules (group of densely connected nodes) (Figure 2). Here, we consider binary symmetric input networks. However, similar ideas can be used to weighted graphs. Suppose N is the input network with a modularity matrix M defined as follows:

$$M = N - \frac{dd^T}{2e}, \tag{3}$$

where $d(i)$ is the degree of node i and e is the total number of edges of the network N.

Suppose λ_i is the i-th positive eigenvalue of the matrix M and u_i is its corresponding eigenvector. Suppose the modularity matrix M has p positive eigenvalues. For each node i in the network N, we define a *modularity vector* m_i as follows:

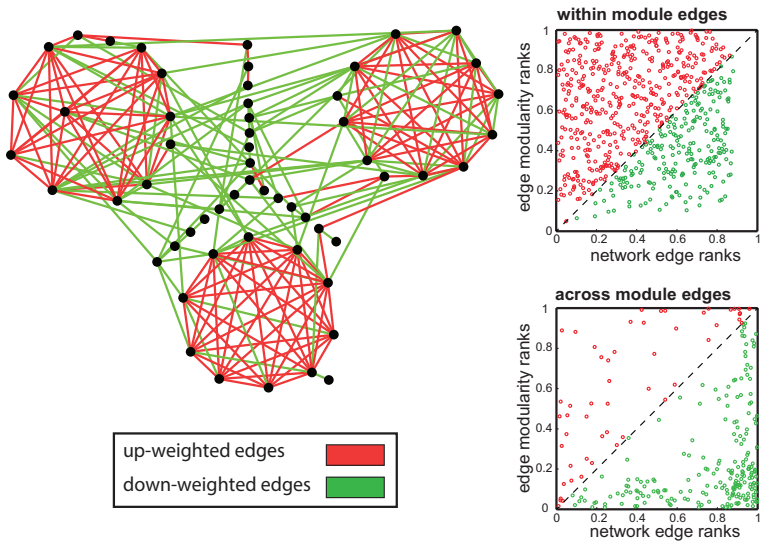

Fig. 2. A spectral method to highlight modules in the network
This figures illustrates how the spectral network modularity algorithm highlights modules over the network.

$$m_i(j) = \sqrt{\lambda_i} u_i(j) \quad 1 \leq j \leq p. \tag{4}$$

Algorithm 1. *A spectral network modularity algorithm can be summarized as follows:*

- *compute node modularity vectors.*
- *compute edge modularity scores by ranking all pairwise distances between node modularity vectors.*
- *form a network based on edge modularities.*

This method assigns higher weights to edges that are more likely to belong to regulatory modules by considering the entire network topology and exploiting spectral analysis of input networks.

To characterize modularity of an input network N, we use a probabilistic approach which compares the modularity of the network with the one of a randomly generated network with the same number of edges. Suppose we have 2^c modules (node partitions) in the network and a $n \times c$ binary matrix $S = (s_1 | \ldots | s_c)$ characterizes those partitions (modules). Two nodes are in the same module, if their corresponding rows in the matrix S are identical.

For a given input network N, network modularity can be defined as follows:

$$Q = (\text{number of edges within modules}) - (\text{expected number of edges across modules}). \quad (5)$$

A probabilistic background network is modeled by assuming that an edge exists between nodes i and j with probability $d(i)d(j)/(2e)$ where $d(i)$ and $d(j)$ are degrees of nodes i and j, and e is the total number of network edges. Under this probabilistic background model, the expected total number of network edges is equal to e since

$$\frac{1}{2}\sum_{i,j}\frac{d(i)d(j)}{2e} = \frac{1}{4e}\sum_{i} d(i)\sum_{j} d(j) = e.$$

Note that, under this model it is more likely to have an edge between two high degrees nodes than between two low degree ones. By using this probabilistic background model, network modularity can be written as follows:

$$\begin{aligned} Q &= \sum_{i,j}^{n}\sum_{k=1}^{c} N(i,j)S(i,k)S(j,k) - \frac{d(i)d(j)}{2e}S(i,k)S(j,k) \\ &= \sum_{i,j=1}^{n} M(i,j)S(i,k)S(j,k), \end{aligned} \quad (6)$$

where M is the modularity matrix defined as in equation (3).

By decomposition of the modularity matrix M to its eigenvalues λ_i and eigenvectors u_i, equation (6) can be re-written as

$$\begin{aligned} Q &= \sum_{i=1}^{n}\sum_{k=1}^{c} \lambda_i (u_i^T s_k)^2 \\ &= \sum_{i=1}^{n}\sum_{k=1}^{c} (\sqrt{\lambda_i} u_i^T s_k)^2, \end{aligned} \quad (7)$$

where s_k is the k-th column of the matrix S. If S were unconstrained (not necessarily a binary matrix), Q would be maximized by choosing columns of S parallel to the most positive eigenvectors of the modularity matrix M. An unconstrained S can be viewed as a soft partitioning of network nodes in contrast to a binary one which is a hard partitioning of network nodes. Therefore, if two nodes i and

j are more likely to be in the same module, their corresponding rows in the matrix S will be close to each other (in the binary case, rows of the matrix S which correspond to nodes in the same module are in fact identical). Having this intuition, we define node modularity vectors as in equation (4) which also assigns higher weights to eigenvectors with larger eigenvalues. We then rank all edges based on distances between their node modularity vectors.

For non-binary networks, first we map all edge weights to be between 0 and 1. To form the modularity matrix M of equation (3), we use vector d whose i-th component is the sum of weights of edges connected to node i (in the binary case, this simplifies to the degree of node i).

The computational complexity of the decomposition step of the proposed spectral network integration algorithm is in the order of $\mathcal{O}(n^3)$, where n is the number of nodes in the network. To reduce the computation cost specially for large networks, one can only use the top q eigenvectors and eigenvalues of the modularity matrix to form node modularity vectors.

IV. Network Deconvolution- A Spectral Method to Distinguish Direct Dependencies over Networks

Network deconvolution introduced in [3] is a systematic approach of computing direct dependencies in a network by use of local edge weights. Suppose G_{obs} represents an observed dependency matrix, a properly scaled similarity matrix among variables. The linear scaling depends on the largest absolute eigenvalue of the un-scaled similarity matrix and is discussed in more details in Section IV-B. Components of G_{obs} can be derived by use of different pairwise similarity metrics such as correlation or mutual information. In particular, $g^{obs}_{i,j}$, the (i,j)-th component of G_{obs}, represents the similarity value between the *observed* patterns of variables i and j in the network.

A perennial challenge to inferring networks is that, both direct and indirect dependencies arise together. A direct information flow modeled by an edge in G_{dir} can give rise to two or higher level indirect information flows captured in G_{indir}:

$$G_{indir} = G^2_{dir} + G^3_{dir} + \dots \tag{8}$$

The power associated with each term in G_{indir} corresponds to the level of indirection contributed by that term. We assume that the observed dependency matrix, G_{obs}, comprises both direct and indirect dependency effects (i.e., $G_{obs} = G_{dir} + G_{indir}$). Further, we assume that G_{dir} is an $n \times n$ decomposable matrix.

Under the modeling assumptions, the following network deconvolution algorithm finds an optimal solution for direct dependency weights by using the observed dependencies:

Algorithm 2. *Network deconvolution has three steps:*

- **Linear Scaling Step:** *The observed dependency matrix is scaled linearly so that all eigenvalues of the direct dependency matrix are between* -1 *and* 1.
- **Decomposition Step:** *The observed dependency matrix* G_{obs} *is decomposed to its eigenvalues and eigenvectors such that* $G_{obs} = U\Sigma_{obs}U^{-1}$.
- **Deconvolution step:** *A diagonal eigenvalue matrix* Σ_{dir} *is formed whose* i*-th component is* $\lambda_i^{dir} = \frac{\lambda_i^{obs}}{\lambda_i^{obs}+1}$. *Then, the output direct dependency matrix is* $G_{dir} = U\Sigma_{dir}U^{-1}$.

A. Optimality analysis of network deconvolution

In this section, we show how the network deconvolution algorithm proposed in Algorithm 2 finds an optimal solution for direct dependency weights by using the observed ones.

Suppose U and Σ_{dir} represent eigenvectors and a diagonal matrix of eigenvalues of G_{dir}, where λ_i^{dir} is the i-th diagonal component of the matrix Σ_{dir}. Also, suppose the largest absolute eigenvalue of G_{dir} is strictly smaller than one. This assumption holds by using a linear scaling function over the unscaled observed dependency network and is discussed in Section IV-B. By using the eigen decomposition principle, we have $G_{dir} = U\Sigma_{dir}U^{-1}$. Therefore,

$$
\begin{aligned}
G_{dir} + G_{indir} &\overset{(a)}{=} G_{dir} + G_{dir}^2 + \dots \\
&\overset{(b)}{=} (U\Sigma_{dir}U^{-1}) + (U\Sigma_{dir}^2U^{-1}) + \dots \\
&= U\big(\Sigma_{dir} + \Sigma_{dir}^2 + \dots\big)U^{-1} \\
&= U\begin{pmatrix} \sum_{i\geq 1}(\lambda_1^{dir})^i & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sum_{i\geq 1}(\lambda_n^{dir})^i \end{pmatrix}U^{-1} \\
&\overset{(c)}{=} U\begin{pmatrix} \frac{\lambda_1^{dir}}{1-\lambda_1^{dir}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\lambda_n^{dir}}{1-\lambda_n^{dir}} \end{pmatrix}U^{-1}.
\end{aligned}
\tag{9}
$$

Equality (a) follows from the definition of diffusion model of equation (8). Equality (b) follows from the

eigen decomposition of matrix G_{dir}. Equality (c) uses geometric series to compute the infinite summation in a closed-form since $|\lambda_i^{dir}| < 1$ for all i.

By using the eigen decomposition of the observed network, G_{obs}, we have $G_{obs} = U\Sigma_{obs}U^{-1}$, where

$$\Sigma_{obs} = \begin{pmatrix} \lambda_1^{obs} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n^{obs} \end{pmatrix}. \tag{10}$$

Therefore, from equations (9) and (10), if

$$\frac{\lambda_i^{dir}}{1-\lambda_i^{dir}} = \lambda_i^{obs} \quad \forall 1 \leq i \leq n, \tag{11}$$

the error term $G_{obs} - \big(G_{dir} + G_{indir}\big) = 0$. Rewriting equation (11) leads to a non-linear filter over eigenvalues in the network deconvolution algorithm:

$$\lambda_i^{dir} = \frac{\lambda_i^{obs}}{1+\lambda_i^{obs}} \quad \forall 1 \leq i \leq n. \tag{12}$$

This nonlinear filter over eigenvalues allows network deconvolution to eliminate indirect dependency effects modeled as in equation (8) and compute a globally optimal solution for direct dependencies over the network with zero error.

B. Scaling effects

As discussed in Section IV-A, to have convergence of the right-hand side of equation (8), the largest absolute eigenvalue of the direct dependency matrix G_{dir} (say β) should be strictly smaller than one (i.e., $\beta < 1$). However, this matrix is unknown in advance and in fact, the whole purpose of having the described model in equation (8) is to compute the direct dependency matrix. In this section, we show that by linearly scaling the un-scaled observed dependency matrix G_{obs}^{us}, it can be guaranteed that the largest absolute eigenvalue of the direct dependency matrix is strictly less than one. In the following, we describe how the linear scaling factor α is chosen where $G_{obs} = \alpha G_{obs}^{us}$.

Suppose $\lambda_+^{obs(us)}$ and $\lambda_-^{obs(us)}$ are the largest positive and smallest negetive eigenvalues of G_{obs}^{us}. Then, by having

$$\alpha \leq \max\big(\frac{\beta}{(1-\beta)\lambda_+^{obs(us)}}, \frac{-\beta}{(1+\beta)\lambda_-^{obs(us)}}\big), \tag{13}$$

the largest absolute eigenvalue of G_{dir} will be less or equal to $\beta < 1$.

In the following, we show how inequality (13) is derived. Say λ^{dir} and $\lambda^{obs(us)}$ are eigenvalues of direct and unscaled observation matrices. By a similar argument as the one of equation (9), we have

$$\lambda^{dir} = \frac{\lambda^{obs(us)}}{\frac{1}{\alpha} + \lambda^{obs(us)}}. \tag{14}$$

We consider three cases: $\lambda^{obs(us)} \geq 0$, $-1/\alpha < \lambda^{obs} < 0$ and $\lambda^{obs(us)} < -1/\alpha$ (in the case of $\lambda^{obs(us)} = -1/\alpha$, the function is undefined. Hence, we choose α to avoid this case).

- **Case 1:** when $\lambda^{obs(us)} \geq 0$: In this case, we have

$$\frac{\lambda^{obs(us)}}{\frac{1}{\alpha} + \lambda^{obs(us)}} \leq \beta < 1 \Rightarrow \alpha \leq \frac{\beta}{(1-\beta)\lambda^{obs(us)}}$$

- **Case 2:** when $-1/\alpha < \lambda^{obs} < 0$: In this case, we have

$$\frac{-\lambda^{obs(us)}}{\frac{1}{\alpha} + \lambda^{obs(us)}} \leq \beta < 1 \Rightarrow \alpha \leq \frac{-\beta}{(1+\beta)\lambda^{obs(us)}}$$

- **Case 3:** when $\lambda^{obs(us)} < -1/\alpha$:

  In this case, we have

$$\frac{\lambda^{obs(us)}}{\frac{1}{\alpha} + \lambda^{obs(us)}} \leq \beta < 1 \Rightarrow \frac{1}{\alpha} \leq \frac{1-\beta}{\beta}\lambda^{obs(us)} < 0,$$

  which is not possible since $\alpha > 0$. Therefore, α should be chosen so that eigenvalues of the unscaled observation matrix do not fall in this regime. In other words, for negative $\lambda^{obs(us)}$, $\alpha < \frac{-1}{\lambda^{obs(us)}}$. This condition trivially holds as a consequence of Case 2.

Putting all three cases together, inequality (13) is derived which guarantees the largest absolute eigenvalue of the direct dependency matrix is less than or equal to $0 < \beta < 1$.

In equation (8), diffusion effects decay exponentially with respect to the indirect path length. For example, second order indirect effects decay proportional to β^2 where β is the largest absolute eigenvalue of G_{dir}. Therefore, smaller β means faster convergence of the right-hand side of equation (8) and in turn faster decay of diffusion effects. In other words, if β is very small, higher order interactions play insignificant roles in observed dependencies since they decay proportionally to β^k where k is the order of indirect interaction.

REFERENCES

[1] M. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[2] Farkas, I., I. Dernyi, H. Jeong, Z. Neda, Z. N. Oltvai, E. Ravasz, A. Schubert, A-L. Barabsi, and T. Vicsek. Networks in life: Scaling properties and eigenvalue spectra. Physica A: Statistical Mechanics and its Applications 314.1 (2002): 25-34.

[3] Soheil Feizi, Daniel Marbach, Muriel Médard and Manolis Kellis Network Deconvolution - A General Method to Distinguish Direct Dependencies over Networks. Nature biotechnology.