Inference of gene networks from temporal gene expression profiles

M. Bansal and D. di Bernardo

Abstract: Genes interact with each other in complex networks that enable the processing of information and the metabolism of nutrients inside the cell. A novel inference algorithm based on linear ordinary differential equations is proposed. The algorithm can infer the local network of gene–gene interactions surrounding a gene of interest from time-series gene expression profiles. The performance of the algorithm has been tested on in silico simulated gene expression data and on a nine gene subnetwork part of the DNA-damage response pathway (SOS pathway) in the bacteria Escherichia coli. This approach can infer regulatory interactions even when only a small number of measurements is available.

1 Introduction

Genes interact with each other in complex networks that enable the processing of information and the metabolism of nutrients inside the cell. Identification of gene regulatory networks is essential for understanding cellular functions. Understanding such networks requires simultaneous measurements of thousands of molecules in the cells. Gene expression microarrays are a recent high-throughput technology that allows measuring quantitatively the expression level of each of the thousands of genes in a cell at a specific condition and time. Molecular biology is therefore rapidly evolving into a quantitative science, and as such, it is increasingly relying on engineering, applied physics and mathematics to make sense of quantitative high-throughput data.

In molecular biology experiments often involve the perturbation of gene of interest (i.e. overexpression or knockdown) and the measurement of the changes in expression level of all the genes at different time points following the perturbation. The focus of computational systems biology is to infer, or ‘reverse-engineer’, from this gene expression microarray data, the regulatory interactions among genes.

Recently, there have been many attempts to reverse engineer gene networks from gene expression data that are based on different mathematical models. These models are mainly classified in four broad classes: (i) Boolean networks, (ii) Bayesian networks, (iii) Information-theoretic approaches and (iv) Differential equations.

Boolean networks [1–3] offer a binary, discrete-time description of a system. This model assumes that each gene is in two possible states, expressed or not expressed. Interactions among genes are modelled through Boolean logic functions. These models are quite simple and can be easily applied but the underlying assumption seems very unrealistic, in particular, modelling genes as discrete entities assuming one of only two states. Also the amount of data needed to infer the network increases as 2^n, where n is the number of genes.

A Bayesian network [4, 5] is a graphical model for probabilistic relationships among a set of random variables X_i, where i = 1, . . . , n. These relationships are encoded in the structure of a directed acyclic graph G, whose vertices (or nodes) are the random variables X_i. The relationships between the variables are described by a joint probability distribution P(X_1, . . . , X_n) that is consistent with the independence assertions embedded in the graph G. These models are good to capture the stochastic nature of gene regulatory network and they can also deal with noisy measurements. Learning Bayesian networks from data is NP hard; hence heuristic search methods have to be used, which do not guarantee convergence to the global optimal solution.

Information-theoretic approaches [6–8], use a generalisation of pairwise correlation coefficient called mutual information (MI), to compare expression profiles from a set of microarrays. For each pair of genes (i, j), their MI_i,j is computed and the edge is set to 0 or 1 depending on a significant threshold to which MI_i,j is compared. MI measures the degree of independence between two genes. The computation of MI requires each data point, that is, each experiment, to be statistically independent from others. Thus it can only deal with steady-state gene expression data set and not with time series.

Reverse-engineering algorithms based on ordinary differential equations [9–13] provide a deterministic approach which is based on relating changes in gene transcript concentration to each other. To reverse-engineer a network using ODE requires a choice of functional form and nonlinear functions can lead to exponential rise in the unknown parameters to be estimated. Higher noise in the microarray data limits their application to make only qualitative statement and not quantitative statement about the underlying network.

Current methods [14] use steady-state expression measurements following multiple perturbations and achieve a very good performance. However, reverse engineering from time-series experiments is still problematic [14] because of the limited number of datapoints that is possible to collect. Only few of the methods have shown some
success with time series data [4, 5, 13]. Yu et al. [5] recovered the network among 20 nodes in the network but to reach a good accuracy they used number of datapoints order of magnitude higher than the number of nodes in network. van Someren et al. [13] used only 11 datapoints to recover the network among 101 genes but their validation was limited to the co-citation of the genes in recovered connection in literature. Dojer et al. [4] successfully recovered the network among 10 genes but they used multiple perturbation experiments with many time points in each experiment.

In this work, we propose a novel method, based on linear differential equations, able to achieve a good performance even for a short time-series gene expression profile coming from a single perturbation experiment.

2 Methods

2.1 Network model description

Our model is based on relating the changes in gene transcription to each other and to the external perturbation (Fig. 1). By external perturbation we mean an experimental treatment that can alter the transcription rate of the genes in the cell. An example of perturbation is the treatment with a chemical compound (i.e. a drug), or a genetic perturbation involving overexpression or downregulation of particular genes. We used a linear system of ordinary differential equations [15] to represent the rate of synthesis of a transcript as a function of the concentrations of every other transcript in a cell and the external perturbation

\[
\dot{x}_i(t_k) = \sum_{j=1}^{N} a_{ij} x_j(t_k) + \sum_{l=1}^{P} b_{il} u(t_k)
\]

where \(i = 1, \ldots, N; k = 1, \ldots, M; x(t_k)\) is concentration of transcript \(i\) measured at time \(t_k; \dot{x}_i(t_k)\) is the rate of change of concentration of gene transcript \(i\) at time \(t_k\), that is the first derivative of the mRNA concentration of gene \(i\) measured at time \(t_k; a_{ij}\) represents the influence of gene \(j\) on gene \(i\) with a positive, zero or negative sign indicating activation, no interaction and repression, respectively; \(b_{il}\) represents the effect of \(l\)th perturbation on \(x_i\) and \(u(t_k)\) represents the \(l\)th external perturbation at time \(t_k\).

Equation (1) at time \(t_k\) can be rewritten in a more compact form using matrix notation

\[
X(t_k) = AX(t_k) + BU(t_k) \quad k = 1, \ldots, M
\]

where \(X(t_k)\) is an \(N \times 1\) vector of mRNA concentration of \(N\) genes at time \(t_k\), \(X(t_k)\) is an \(N \times 1\) vector of the first derivatives of \(X\) at time \(t_k, A\) is an \(N \times N\) connectivity matrix, composed of elements \(a_{ij}, B\) is an \(N \times P\) matrix representing the effect of \(P\) perturbations on \(N\) genes, \(U(t_k)\) is \(P \times 1\) vector representing \(P\) perturbations at time \(t_k\) and \(M\) is the number of time points \(t_k\) in the time-series experiment.

The unknown connectivity matrix \(A\) is what we want to learn from the data. Element \((i, l)\) of \(B\) will be different from zero if the \(i\)th gene is a direct target of the \(l\)th perturbation.

2.2 Algorithm

To solve (2), we need to compute the derivative of \(X(t_k)\). Since gene expression measurements are noisy, computing derivatives is not appropriate since this will amplify the noise in the data. To avoid this problem, we integrated (2)

\[
\int_0^{t_k} \dot{X}(t) dt = \int_0^{t_k} X(t) dt + B \int_0^{t_k} U(t) dt, \quad k = 1, \ldots, M
\]

or

\[
X(t_k) = A \int_0^{t_k} X(t) dt + B \int_0^{t_k} U(t) dt, \quad k = 1, \ldots, M
\]

where we assumed null initial conditions \([X(0) = 0]\).

Gene expression levels are measured at discrete time points, therefore assuming a ZOH (zero-order-hold) model we can rewrite the (4) as

\[
X(t_k) = A X(t_k) * \delta t + B S^k \delta t
\]

where \(\delta t\) is the sampling interval. This equation can be written as

\[
Y(t_k) = A * H(t_k)
\]

where \(Y(t_k) = X(t_k) - B S^k \delta t\) and \(H(t_k) = S^k \delta t\).

Writing (6) in matrix form after removing time subscript yields

\[
Y = A * H
\]

where \(Y\) and \(H\) are \(N \times M\) dimensional. In what follows, we will use the functional expanded form for each row \(i\) and column \(j\), \(Y_{ij} = f^{a_{ij}}(H_{ij})\) which is equivalent to \(Y_{ij} = \sum_{l=1}^{P} a_{ij} H_{ij}\).

To identify the network means to compute matrix \(A\) from the available data \(Y\) and \(H\). This can be achieved by applying the pseudo-inverse method to each row of \(A\) to obtain

\[
A_i = Y_i * H^i * (H^i H^* + \lambda_i I)^{-1}
\]

where \(A_i\) is the \(i\)th row of \(A, Y_i\) is the \(i\)th row of \(Y, H^i\) is the transpose of \(H, I\) is the identity matrix and \(\lambda_i\) is the ridge parameter. \(\lambda_i\) is computed as the smallest singular value of \([H Y^*]\) and is useful to control for bad conditioning of \(H^* H^*\) [16]. Equation (8) can be solved only if \(M \geq N\), that is if we have a sufficient number of data points. Usually in biology \(M \leq N\).

![Fig. 1 Methodology](image)
Generally gene networks are not fully connected but sparse, that is each gene regulates only a limited number of genes [10]. Therefore we can solve (8) assuming that only \( K \) out of \( N \) parameters of row \( A \), are different from zero. Further assuming that \( K < M \), ensures that we can always solve (8). The problem is to select which \( K \) parameters to use out of all the possible \( K \) over \( N \) combinations.

We applied a forward step-wise regression method [17] where we first considered all solutions with a single parameter \( K = 1 \). We then ranked the solutions according to an error (i.e. least square error) and picked the \( D \) solutions with the smallest error. For each one of the \( D \) solutions, we then consider all the possible pairwise combinations of \( K = 2 \) parameters that contained the previously chosen parameter. This can be repeated up to \( K = M \) connections. We chose \( D \) equal to 5 as suggested in [18] as a good compromise between computation time and accuracy.

### 2.3 Time-series network identification using bootstrapping

At each step, we could have used the least-squared error as the function to minimise. But least-squared error may cause the computed solution to be very sensitive to noise in the data (over-fitting). Bootstrapping is a general technique to avoid overfitting. The basic idea is to randomly draw data (over-fitting). Bootstrapping is a general technique to

\[
\hat{\omega} = \frac{0.632}{1 - 0.632\hat{R}}
\]

\( \hat{\omega} \) ranges from 0.632 if \( \hat{R} = 0 \) to 1 if \( \hat{R} = 1 \). It produces a compromise between the leave-one-out bootstrap and the training error rate that depends on the amount of overfitting. \( \hat{R} \) is called ‘relative overfitting rate’ defined by

\[
\hat{R} = \frac{\hat{\text{Err}} - \text{err}}{\gamma - \text{err}}
\]

\( \gamma \) is called no-information error rate which is the error rate of our prediction rule if input and output are independent, It is computed as

\[
\gamma = \frac{1}{M^2} \sum_{j=1}^{M} \sum_{l=1}^{M} (Y_{ij} - f^{\hat{a}_b}(H_{ij}))^2
\]

We then used \( \hat{\text{Err}}^{0.632+} \) to select the best \( D \) solutions at each step. We used this best \( D \) solutions for next step of forward stepwise regression. At the end of next step, if we found that the minimum \( \hat{\text{Err}}^{0.632+} \) is more than the minimum \( \hat{\text{Err}}^{0.632+} \) of previous step, we stopped the forward stepwise regression and considered the solution from previous step as best solution. At the end, we were left with a unique combination of \( I \) (= \( K \) if forward stepwise method runs until the end, or less if forward step-wise method stops earlier) out of \( N \) parameters for each row \( \mathbf{A}_i \); however, we had \( G \) different estimates of the parameters \( a_{ib} \). We used ‘bagging’ [16] to select a unique solution. Bagging simply chooses the \( a_{ib} \) estimate that minimises the least-square error \( \text{Err}_g \)

\[
g: \arg \min_{g} \{ \text{Err}_g = \frac{1}{M} \sum_{j=1}^{M} (Y_{ij} - f^{\hat{a}_b}(H_{ij}))^2 \}
\]

### 2.4 Time-series network identification using final prediction error

We also tested another method for model selection that we named as time-series network identification using final prediction error (TSNIF). This method is based on computing Akaike’s Final Prediction error (FPE) [19] which is defined as

\[
\text{FPE} = \frac{M + l}{M - l} \text{err}
\]

where err is same as defined in (11), \( M \) is the number of time points and \( l \) is the number of parameters in the model. According to Akaike’s theory, the most accurate model has the smallest FPE. To achieve this, at each step of forward stepwise regression we computed FPE and moved to next the step in the iteration, until we did not find further improvement in FPE. Final solution is obtained from least-square method for the corresponding \( l \) which gives the minimum FPE.

### 3 Data set

#### 3.1 Simulated gene expression data

To test the performance of the algorithm, we generated random networks of \( N \) genes with an average of \( K \) connections per gene. We produced two sets of simulations choosing \( N = 10 \) with \( K = 2 \) for the first set of 20 random networks, and \( N = 20 \) with \( K = 5 \) for the second set of 20
random networks. To generate each random network, we first created a random network using MATLAB® command rss which generates stable continuous-time state-space models \((i = Ax + Bu; y = Cx + Du)\). rss command needs as inputs: the order of model, number of outputs \((y)\) and the number of inputs \((u)\) to the system. rss generates four matrices, \(A, B, C, D\) which all together define the state-space models [19].

The algorithm which rss uses to generate state-space model is as follows. For the network of size \(N\), first it randomly chooses the number of complex and real eigenvalues for matrix \(A\) (complex eigenvalues always exist in pairs, eigen-value and its complex conjugate). For real part of eigen-values, negative random numbers generated from normal distribution are taken and for imaginary part positive random numbers (again from normal distribution) are taken. These eigen-values are placed on the diagonal of a zero matrix, \(Z\), of order \(N \times N\). If the eigen-value pair is complex, then the real and imaginary part of the eigen-values and its complex conjugate are placed on block diagonal of \(Z\). Another matrix, \(T\), of size \(N \times N\) is generated randomly from normal distribution and is then orthogonalised. Finally \(A\) is computed as \(T^T Z T\). \(B, C, D\) and \(D\) are generated randomly using normal distribution.

Once we generated state-space model, we set \(C\) equal to identity matrix and \(D\) equal to a zero matrix so that the output of the system equals its states. Here \(A\) represents the network which is a full-rank matrix with eigen-values whose real part are less than 0 to ensure the stability of the dynamical system [19], that is, all the gene mRNAs reach an equilibrium between their transcription rate and degradation rate after a given time period. For each row of \(A\), we then randomly selected \(K\) elements (including the diagonal), and set the other elements to zero (to make the network sparse). We then checked the stability of this new network by computing its eigen-values, and if \(A\) was not stable we selected a different set of \(K\) elements in each row unless we got a stable network \(A\).

We applied a single perturbation to a single gene of each random network \(A\). The information of the gene that is perturbed is contained in matrix \(B\) \((N \times 1)\). \(B\) has all its elements equal to 0 except for the gene that is perturbed, which is chosen randomly. \(U\) \((1 \times M)\) contains the information about what kind of perturbation is applied. In our simulations, we applied a constant step perturbation of amplitude equal to 1. Once matrix \(A\) \((N \times N)\) and \(B\) \((N \times 1)\) were generated, we simulated the gene expression profile data set \(X = [X(t_1), \ldots, X(t_M)]\) of equally sampled time points which was obtained using lsim command in MATLAB® [10]. Time \(t_M\) was chosen equal to four times the inverse of the smallest eigen-value of \(A\) [19]. This ensures that at time \(t_M\) all the genes are close to their steady-state value. We tested the algorithm performance for increasing number of time points \((M = 10, 20, 40\) and 100 for 10 gene network and \(M = 20, 40\) and 100 for 20 gene network). White Gaussian noise was added to the data matrix with zero mean and standard deviation \(\sigma = \alpha \|X\|\) where \(\|X\|\) represents the absolute values of the elements of \(X\) and \(\alpha\) is chosen as 0, 0.1 and 0.3. \(\alpha = 0\) refers to data with no noise [10].

### 3.2 Experimental data set

We also tested our algorithm on a nine gene network in *Escherichia coli*. Nine genes selected are part of the DNA-damage response pathway (SOS pathway). Gene expression profiles for the nine genes were measured at six different points (with three replicates) following treatment with Norfloxacin, a known antibiotic that acts by damaging the DNA. These nine genes were selected, since the network among them is well known in the literature. For more details, refer to the work Bansal *et al.* [20].

### 4 Results

In order to check the performance of the algorithm for each simulated data set, we computed positive predictive value (PPV) or accuracy defined as \(TP/(TP + FP)\) and sensitivity (Se) defined as \(TP/(TP + FN)\) with TP: true positive; FP: false positive and FN: false negative. Before checking its performance, we rounded all the values of recovered networks to the second decimal place and removed the self-feedback (diagonal elements from \(A\)) from all recovered and original networks. The reason being not to give an advantage to our algorithm, since it always recovers diagonal elements. Our algorithm infers the network as a signed directed graph (Fig. 2), but we checked its performance on recovering also the directed graph (no sign). We computed: a signed directed graph PPV\(^s\) by defining a TP when \(\text{sign}(\hat{a}_{ij}) = \text{sign}(a_{ij})\) and \(a_{ij} \neq 0\), \(\hat{a}_{ij} \neq 0\), that is the sign of the recovered parameter equals the sign of the original parameter and both are different from zero; a directed graph PPV\(^d\) by defining a TP when \(a_{ij} \neq 0\), \(\hat{a}_{ij} \neq 0\); that is both are different from 0 and similarly we computed Se\(^s\) and Se\(^d\) [14].

We also computed a Random PPV, that is the PPV obtained by randomly selecting the same number of genes as in the original network. To check if any algorithm is significantly better than random on a given data or not, we computed the p-value using a binomial distribution.

![Fig. 2  Example of signed network](image)

Dashed line represents negative connection and straight line represents positive connection

- **a** Original network
- **b** Network recovered by TSNiB for 10% noise level in the data
- **c** Network recovered by TSNiF for 10% noise level in the data
4.1 Simulated data

We applied time-series network identification using bootstrapping (TSNIB) and TSNIF on all data sets using the information of perturbation (U) and which gene is perturbed (B). For forward stepwise regression step, we iterated till $K = 2$ and $K = 5$ for 10 and 20 gene networks, respectively. We computed the final PPV and Se for each data set by taking the mean and standard error mean (SEM) of PPV and Se of 20 networks for each data set. Results for the 10 gene network data set are shown in Table 1 and for 20 gene network data set in Table 2. SEM for PPV and Se is not shown in the table for clarity but is always found to be in range 0.00 and 0.04 for all data sets. Bold numbers in table are the results that are significantly better than random PPV with P-value less than 0.1.

Result shows that both TSNIB and TSNIF method can partially recover the real network. Their accuracy (PPV) decreases with increasing noise in the data, but increasing the number of time points used to infer the network can compensate for this. This result is consistent for both 10 and 20 gene network data sets. Performance of TSNIB and TSNIF are similar for 10 gene network but for 20 gene network, for noisy data set, PPV of TSNIF is higher than the PPV of TSNIB method but Se is lower and vice versa. So TSNIF is better in getting small number of connection with higher accuracy but TSNIB recovers more connections with less accuracy compared to TSNIF. Also we observed that both these methods are better in recovering signed networks than directed networks. This can be explained as both these algorithms are designed to recover signed networks. So whenever they recover any correct connection, they usually recover it with correct sign. Time needed on INTEL XEON 3 GHz processor for TSNIB for 10 gene network ranges from 1 to 10 min for increasing number of data points and for 20 gene network it is 30 min to 2 h. For TSNIF, it is almost 1 s for 10 gene network and 5 s for 20 gene network for all number of data points. TSNIB method is computational intensive and is not scalable. TSNIF is scalable and is possible to apply on large networks of order 1000 genes.

4.2 Experimental data

We applied both TSNIB and TSNIF on a nine gene subnetwork, part of SOS network in E. coli. To this end, we computed the average of the three replicates for each time point following treatment with Norfloxacin. As the algorithm requires the first time point to be zero, we computed the log2 (log base 2) ratio of all time points with respect to the SOS pathway via activation of the recA protein. So we considered $B = 1$ only for the element corresponding

Table 1: Results of the application of network inference algorithms on 20 networks of 10 genes

<table>
<thead>
<tr>
<th>Number of data points</th>
<th>$\alpha$</th>
<th>TSNIB PPV</th>
<th>TSNIB Se</th>
<th>TSNIF PPV</th>
<th>TSNIF Se</th>
<th>TSNIF* PPV</th>
<th>TSNIF* Se</th>
<th>Banjo PPV</th>
<th>Banjo Se</th>
<th>Random PPV</th>
<th>Random Se</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0.39</td>
<td>0.36</td>
<td>0.39</td>
<td>0.36</td>
<td>0.26</td>
<td>0.28</td>
<td>0</td>
<td>0</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.37*</td>
<td>0.34*</td>
<td>0.37*</td>
<td>0.34*</td>
<td>0.24*</td>
<td>0.26*</td>
<td>0</td>
<td>0</td>
<td>0.10*</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>0.29</td>
<td>0.27</td>
<td>0.32</td>
<td>0.26</td>
<td>0.25</td>
<td>0.27</td>
<td>0.31</td>
<td>0.05</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.25*</td>
<td>0.23*</td>
<td>0.29*</td>
<td>0.23*</td>
<td>0.23*</td>
<td>0.25*</td>
<td>0.19</td>
<td>0.04</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.3</td>
<td>0.26</td>
<td>0.24</td>
<td>0.32</td>
<td>0.24</td>
<td>0.23</td>
<td>0.25</td>
<td>0.30</td>
<td>0.09</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.18*</td>
<td>0.17*</td>
<td>0.26*</td>
<td>0.19*</td>
<td>0.19*</td>
<td>0.21</td>
<td>0.20</td>
<td>0.06</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0.50</td>
<td>0.47</td>
<td>0.51</td>
<td>0.47</td>
<td>0.23</td>
<td>0.24</td>
<td>0.10</td>
<td>0.02</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.48*</td>
<td>0.45*</td>
<td>0.50*</td>
<td>0.46*</td>
<td>0.21*</td>
<td>0.22*</td>
<td>0.08</td>
<td>0.01</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.1</td>
<td>0.34</td>
<td>0.32</td>
<td>0.35</td>
<td>0.28</td>
<td>0.25</td>
<td>0.26</td>
<td>0.23</td>
<td>0.09</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.26</td>
<td>0.29*</td>
<td>0.29*</td>
<td>0.24</td>
<td>0.22*</td>
<td>0.23*</td>
<td>0.19</td>
<td>0.07</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.3</td>
<td>0.32</td>
<td>0.29</td>
<td>0.31</td>
<td>0.25</td>
<td>0.24</td>
<td>0.25</td>
<td>0.22</td>
<td>0.17</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.22</td>
<td>0.20*</td>
<td>0.24*</td>
<td>0.19*</td>
<td>0.21*</td>
<td>0.22*</td>
<td>0.14</td>
<td>0.11</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>0.62</td>
<td>0.61</td>
<td>0.61</td>
<td>0.60</td>
<td>0.29</td>
<td>0.28</td>
<td>0.06</td>
<td>0.01</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.60*</td>
<td>0.59*</td>
<td>0.60*</td>
<td>0.59*</td>
<td>0.28*</td>
<td>0.26*</td>
<td>0.05</td>
<td>0.00</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.1</td>
<td>0.37</td>
<td>0.35</td>
<td>0.36</td>
<td>0.30</td>
<td>0.29</td>
<td>0.28</td>
<td>0.27</td>
<td>0.15</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.30*</td>
<td>0.28*</td>
<td>0.31*</td>
<td>0.26*</td>
<td>0.28*</td>
<td>0.26*</td>
<td>0.20</td>
<td>0.11</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.3</td>
<td>0.30</td>
<td>0.29</td>
<td>0.33</td>
<td>0.25</td>
<td>0.28</td>
<td>0.27</td>
<td>0.15</td>
<td>0.13</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.23*</td>
<td>0.22*</td>
<td>0.27*</td>
<td>0.21</td>
<td>0.26*</td>
<td>0.25*</td>
<td>0.12</td>
<td>0.10</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>0.70</td>
<td>0.70</td>
<td>0.69</td>
<td>0.69*</td>
<td>0.32</td>
<td>0.30</td>
<td>0.05</td>
<td>0.00</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.69*</td>
<td>0.69*</td>
<td>0.69*</td>
<td>0.68*</td>
<td>0.31*</td>
<td>0.29*</td>
<td>0.05</td>
<td>0.00</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.1</td>
<td>0.34</td>
<td>0.33</td>
<td>0.37</td>
<td>0.31*</td>
<td>0.33*</td>
<td>0.31*</td>
<td>0.23</td>
<td>0.11</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.30*</td>
<td>0.29*</td>
<td>0.34*</td>
<td>0.28*</td>
<td>0.31*</td>
<td>0.29*</td>
<td>0.17</td>
<td>0.08</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.3</td>
<td>0.31</td>
<td>0.30</td>
<td>0.36</td>
<td>0.29</td>
<td>0.32</td>
<td>0.30</td>
<td>0.16</td>
<td>0.04</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.25*</td>
<td>0.24*</td>
<td>0.29*</td>
<td>0.24</td>
<td>0.30*</td>
<td>0.28*</td>
<td>0.12</td>
<td>0.03</td>
<td>0.10</td>
<td></td>
</tr>
</tbody>
</table>

PPV, positive predicted value; Se, sensitivity, $\alpha$, noise level. In bold are the results that performed significantly better than random, using as a random model a binomial distribution. *corresponds to performance of TSNI by considering top $K$—sparsity of network) elements in each row not equal to zero and setting rest to zero.
Table 2: Results of the application of network inference algorithms on 20 networks of 20 genes

<table>
<thead>
<tr>
<th>Number of data points</th>
<th>α</th>
<th>TSNIB PPV</th>
<th>TSNIB Se</th>
<th>TSNIF PPV</th>
<th>TSNIF Se</th>
<th>TSNIFa PPV</th>
<th>TSNIFa Se</th>
<th>BANJO PPV</th>
<th>BANJO Se</th>
<th>Random PPV</th>
<th>Random Se</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0</td>
<td>0.33d</td>
<td>0.28d</td>
<td>0.31d</td>
<td>0.28d</td>
<td>0.28d</td>
<td>0.30d</td>
<td>0.16d</td>
<td>0.02d</td>
<td>0.25d</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>0.29d</td>
<td>0.25d</td>
<td>0.32d</td>
<td>0.19d</td>
<td>0.21d</td>
<td>0.26d</td>
<td>0.28d</td>
<td>0.29d</td>
<td>0.07d</td>
<td>0.25d</td>
<td></td>
</tr>
<tr>
<td>0.17</td>
<td>0.15</td>
<td>0.22a</td>
<td>0.13a</td>
<td>0.20a</td>
<td>0.21a</td>
<td>0.26d</td>
<td>0.28a</td>
<td>0.19a</td>
<td>0.05a</td>
<td>0.12a</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>0.27d</td>
<td>0.24d</td>
<td>0.28d</td>
<td>0.19d</td>
<td>0.25d</td>
<td>0.28d</td>
<td>0.24a</td>
<td>0.07d</td>
<td>0.12a</td>
<td>0.25d</td>
<td></td>
</tr>
<tr>
<td>0.14</td>
<td>0.13a</td>
<td>0.16a</td>
<td>0.11a</td>
<td>0.17a</td>
<td>0.17a</td>
<td>0.16a</td>
<td>0.04a</td>
<td>0.12a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>0.35d</td>
<td>0.31d</td>
<td>0.35d</td>
<td>0.32d</td>
<td>0.26d</td>
<td>0.28a</td>
<td>0.16d</td>
<td>0.01d</td>
<td>0.25d</td>
<td></td>
</tr>
<tr>
<td>0.27</td>
<td>0.24a</td>
<td>0.24a</td>
<td>0.22a</td>
<td>0.21a</td>
<td>0.22a</td>
<td>0.26a</td>
<td>0.14a</td>
<td>0.01a</td>
<td>0.12a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>0.30d</td>
<td>0.27d</td>
<td>0.35d</td>
<td>0.16d</td>
<td>0.26d</td>
<td>0.27d</td>
<td>0.26d</td>
<td>0.09d</td>
<td>0.25d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.17</td>
<td>0.16a</td>
<td>0.24a</td>
<td>0.11a</td>
<td>0.21a</td>
<td>0.22a</td>
<td>0.17a</td>
<td>0.06a</td>
<td>0.12a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>0.25d</td>
<td>0.22d</td>
<td>0.31d</td>
<td>0.16d</td>
<td>0.27d</td>
<td>0.28d</td>
<td>0.29d</td>
<td>0.14d</td>
<td>0.25d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.18</td>
<td>0.16a</td>
<td>0.20a</td>
<td>0.11a</td>
<td>0.21a</td>
<td>0.22a</td>
<td>0.18a</td>
<td>0.09a</td>
<td>0.12a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>0.35d</td>
<td>0.32d</td>
<td>0.34d</td>
<td>0.31d</td>
<td>0.28d</td>
<td>0.25d</td>
<td>0.34d</td>
<td>0.02d</td>
<td>0.25d</td>
<td></td>
</tr>
<tr>
<td>0.28</td>
<td>0.25a</td>
<td>0.28a</td>
<td>0.28a</td>
<td>0.25a</td>
<td>0.22a</td>
<td>0.20a</td>
<td>0.26a</td>
<td>0.01a</td>
<td>0.12a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>0.31d</td>
<td>0.28d</td>
<td>0.37d</td>
<td>0.17d</td>
<td>0.27d</td>
<td>0.25d</td>
<td>0.23a</td>
<td>0.05d</td>
<td>0.25d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.20</td>
<td>0.18a</td>
<td>0.29a</td>
<td>0.13a</td>
<td>0.21a</td>
<td>0.20a</td>
<td>0.13a</td>
<td>0.03a</td>
<td>0.12a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>0.30d</td>
<td>0.22d</td>
<td>0.34d</td>
<td>0.16d</td>
<td>0.27d</td>
<td>0.25d</td>
<td>0.18d</td>
<td>0.02a</td>
<td>0.25d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.18</td>
<td>0.16a</td>
<td>0.25a</td>
<td>0.11a</td>
<td>0.22a</td>
<td>0.20a</td>
<td>0.15a</td>
<td>0.01a</td>
<td>0.12a</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PPV, positive predicted value; Se, sensitivity; α, noise level. In bold are the results that performed significantly better than random, using as a random model a Binomial distribution. *Corresponds to performance of TSNI by considering top K (= sparsity of network) elements in each row not equal to zero and setting rest to zero.

Table 3: Results on E. coli data set

<table>
<thead>
<tr>
<th></th>
<th>TSNIB PPV</th>
<th>TSNIB Se</th>
<th>TSNIF PPV</th>
<th>TSNIF Se</th>
<th>TSNIFa PPV</th>
<th>TSNIFa Se</th>
<th>BANJO PPV</th>
<th>BANJO Se</th>
<th>Random PPV</th>
<th>Random Se</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPV</td>
<td>0.65d</td>
<td>0.35d</td>
<td>0.64d</td>
<td>0.21d</td>
<td>0.60d</td>
<td>0.63d</td>
<td>0d</td>
<td>0d</td>
<td>0.6d</td>
<td></td>
</tr>
<tr>
<td>Se</td>
<td>0.52*</td>
<td>0.28*</td>
<td>0.50*</td>
<td>0.16*</td>
<td>0.42*</td>
<td>0.44*</td>
<td>0*</td>
<td>0*</td>
<td>0.3*</td>
<td></td>
</tr>
</tbody>
</table>

PPV, positive predicted value; Se, sensitivity; α, noise level. In bold are the results that performed significantly better than random, using as a random model a Binomial distribution.

*Corresponds to the performance of TSNI when we used the information from [10] that each gene is connected with other five genes (K = 5) and set the four weakest connection in each row of A to zero.

4.3 Comparison with other inference methods

State-of-the-art inference methods (for a review, see [21]) are designed to infer regulatory interactions among genes.

Table 4: Results of the application of network inference algorithms on 20 networks of 20 genes

<table>
<thead>
<tr>
<th></th>
<th>TSNIB PPV</th>
<th>TSNIB Se</th>
<th>TSNIF PPV</th>
<th>TSNIF Se</th>
<th>TSNIFa PPV</th>
<th>TSNIFa Se</th>
<th>BANJO PPV</th>
<th>BANJO Se</th>
<th>Random PPV</th>
<th>Random Se</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPV</td>
<td>0.65d</td>
<td>0.35d</td>
<td>0.64d</td>
<td>0.21d</td>
<td>0.60d</td>
<td>0.63d</td>
<td>0d</td>
<td>0d</td>
<td>0.6d</td>
<td></td>
</tr>
<tr>
<td>Se</td>
<td>0.52*</td>
<td>0.28*</td>
<td>0.50*</td>
<td>0.16*</td>
<td>0.42*</td>
<td>0.44*</td>
<td>0*</td>
<td>0*</td>
<td>0.3*</td>
<td></td>
</tr>
</tbody>
</table>

PPV, positive predicted value; Se, sensitivity; α, noise level. In bold are the results that performed significantly better than random, using as a random model a Binomial distribution.

*Corresponds to the performance of TSNI when we used the information from [10] that each gene is connected with other five genes (K = 5) and set the four weakest connection in each row of A to zero.

we used the information from [10] that each gene is connected to five other genes, hence we set K = 5.

Before computing PPV and Se, we removed the self-feedback and rounded the recovered network to second place of decimal. Results are shown in Table 3. In the literature, 43 connections (apart from self-feedback) are known among the nine genes of SOS pathway. TSNIB and TSNIF recovered 12 and 7 connections with correct sign with an accuracy of 52 and 50%, respectively. The real network is highly connected; therefore the random PPV for directed graph is very high (e.g. for fully connected network the random PPV is 1), so our method failed to give better performance than random PPV for directed graph.

4.3 Comparison with other inference methods

State-of-the-art inference methods (for a review, see [21]) are designed to infer regulatory interactions among genes.
5 Discussion

In this work, we present a new gene network inference approach to partially recover a gene network from few and noisy measurements. Our approach is based on the assumption that gene expressions are the outcome of a linear model. Even if biological systems are non-linear, this assumption still holds if the perturbation is small. Small perturbations will keep the system close to its stable state thus behaving linearly, but larger (or sometime even complex) perturbations can lead to nonlinear behaviour. In that case, our method will not be appropriate for inference of gene networks. If the system is nonlinear and if there are lot of datapoints then Bayesian approach may be an appropriate algorithm to apply on that data set. To improve the performance of our algorithm, we should perturb more than one gene at a time with a more complex dynamics. This, however, is difficult to implement experimentally, since already a single perturbation requires considerable experimental effort. More complex perturbations are necessary for improving inference performance of any algorithm based on time-series gene expression data, but at this stage it is still not practical in experimental mol-bolariations are necessary for improving inference performance of any algorithm based on time-series gene expression data, and noisy measurements. Our approach is based on the assumption that gene expressions are the outcome of a linear model. Even if biological systems are non-linear, this assumption still holds if the perturbation is small. Small perturbations will keep the system close to its stable state thus behaving linearly, but larger (or sometime even complex) perturbations can lead to nonlinear behaviour. In that case, our method will not be appropriate for inference of gene networks. If the system is nonlinear and if there are lot of datapoints then Bayesian approach may be an appropriate algorithm to apply on that data set. To improve the performance of our algorithm, we should perturb more than one gene at a time with a more complex dynamics. This, however, is difficult to implement experimentally, since already a single perturbation requires considerable experimental effort. More complex perturbations are necessary for improving inference performance of any algorithm based on time-series gene expression data, but at this stage it is still not practical in experimental molecular biology. This situation may soon change, thanks to the progress in synthetic biology that will allow construction of inducible systems with more complex dynamics [22].

6 Acknowledgment

This work was supported by ‘Fondazione Telethon’ Grant TDDP17TELB and the ‘Scuola Europea di Medicina Molecolare-SEMM’ Grant for M.B.

7 References