31st Annual International Conference of the IEEE EMBS
Minneapolis, Minnesota, USA, September 2-6, 2009

The Inferelator 2.0: a scalable framework for reconstruction of dynamic regulatory network models.

Aviv Madar, Alex Greenfield, Harry Ostrer, Eric Vanden-Eijnden and Richard Bonneau

Abstract— **Current methods for reconstructing biological networks often learn either the topology of large networks or the kinetic parameters of smaller networks with a well-characterized topology. We have recently described a network reconstruction algorithm, *the Inferelator* 1.0, that given a set of genome-wide measurements as input, simultaneously learns both topology and kinetic-parameters. Specifically, it learns a system of ordinary differential equations (ODEs) that describe the rate of change in transcription of each gene or gene-cluster, as a function of environmental and transcription factors. In order to scale to large networks, in Inferelator 1.0 we have approximated the system of ODEs to be uncoupled, and have solved each ODE using a one-step finite difference approximation. Naturally, these approximations become crude as the simulated time-interval increases.**

Here we present, implement, and test a new Markov-Chain-Monte-Carlo (MCMC) dynamical modeling method, Inferelator 2.0, that works in tandem with Inferelator 1.0 and is designed to relax these approximations. We show results for the prokaryote *Halobacterium* that demonstrate a marked improvement in our predictive performance in modeling the regulatory dynamics of the system over longer time-scales.

I. INTRODUCTION

Learning and characterizing regulatory networks responsible for the remarkable ability of organisms to adapt to changing environment is a key problem in modern biology with applications spanning bioengineering, drug development, and many other biological fields [1]. Transcriptional regulatory networks (RNs) can be modeled as a system of ordinary differential equations (ODEs), describing the rate of change in mRNA concentrations as a function of relevant predictors. We have recently described a network reconstruction algorithm, the Inferelator 1.0 [2], which given: 1) a microarray compendium composed of time-series and steady-state measurements, and 2) prior information, such as lists of coregulated genes (gene-clusters) and putative transcription factors (TFs), learns for each modeled entity—a gene or a gene-cluster—the kinetic parameters in an ODE model. Importantly, the resultant system of ODEs is sparse, i.e. each modeled entity is regulated by only a few predictors as expected from biological RNs.

Manuscript received April 7, 2009; Accepted June 1, 2009.

A. Madar and R. Bonneau are with the Courant institute of Mathematical Sciences and the Center for Genomics & Systems Biology, New York University, New York, NY 10003 USA (phone: (212) 992-9516, fax: (212) 995-4015, email: bonneau@nyu.edu)

A. Greenfield is with the Computational Biology Program, New York University, New York, NY 10013 USA

H. Ostrer is with the Human Genetics Program, New York University School of Medicine, New York, NY 10016 USA

E. Vanden-Eijnden is with the Courant institute of Mathematical Sciences, New York University, New York, NY 10003 USA

In order to scale to genome-wide networks we have made two major simplifications in Inferelator 1.0. First, we have assumed that the predictor levels are constant throughout a time interval, which becomes a crude approximation as the time-interval length increases. Second, we have solved the system of ODEs as an uncoupled system, which is not a realistic model for the underlying RN. In spite of these simplifications we have recently shown that, at least for prokaryotes, RNs learned using Inferelator 1.0 can explain observed mRNA measurements (training-set), as well as predict newly observed mRNA measurements (a large test-set) [2], [3].

Here we focus on improving the long-time-scale dynamical properties of the RN model from Bonneau et. al. [2], [3] (all data used herein are also taken from this prior work). Towards this goal we describe a new method, Inferelator 2.0, that aim to remove or relax the aformentioned simplifications by: 1) allowing predictors to time evolve throughout a time-interval, and 2) solving the system of ODEs as a coupled system. We use this new method to refine RN models resulting from Inferelator 1.0, and show significant improvement in the predictive performance over longer time intervals.

II. PROBLEM SET UP

The dynamical variables available from microarray observations are the mRNA levels of genes,

$$x(t) = (x_1(t), \dots, x_M(t))^T. \tag{1}$$

The data set comes in the form of a $M \times K$ matrix of observations

$$X = \begin{pmatrix} x_1(t_1) & x_1(t_2) & \cdots & x_1(t_K) \\ x_2(t_1) & x_2(t_2) & \cdots & x_2(t_K) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(t_1) & x_M(t_2) & \cdots & x_M(t_K) \end{pmatrix} \tag{2}$$

where $(t_1, t_2, \dots, t_K)$ are the observation times. Note that X excludes steady state observations, which can be used by Inferelator 1.0 and Inferelator 2.0, but are excluded from this work to simplify our description of the method.

Without loss of generality, we can also assume that each variable (i.e. each row in X) has been normalized such that its time-average is 0 and its variance 1, i.e.

$$\frac{1}{K}\sum_{k=1}^{K} x_i(t_k) = 0,\ \frac{1}{K}\sum_{k=1}^{K} x_i^2(t_k) = 1,\ i = 1, \dots, M. \tag{3}$$

Typically it is desirable to compress the dimension of the data-set (using biclustering or gene-clustering) by estimating co-regulated or co-expressed gene groupings.

To this end, we define $y(t) = (y_1(t), \ldots, y_N(t))^T$ from $x(t)$, by setting:

$$y_i(t) = \frac{1}{M_i} \sum_{j=1}^{M_i} x_{i_j}(t), \qquad i = 1, \ldots, N \tag{4}$$

where for each i, $\{i_1, i_2, \ldots, i_{M_i}\} \subset \{1, 2, \ldots, M\}$. This clustering amounts to a shift from genes (X) as the modeled-unit to gene-groupings (Y) as the modeled-unit, and leaves us with a new (smaller) $N \times K$ matrix of observations Y.

We have previously used *cMonkey* [9], a bi-clustering algorithm, to create Y, and have used it to evaluate Inferelator 1.0 [2]. Here we have taken Y directly from [2], enabling direct comparison of the two methods.

III. INFERELATOR 1.0

The Inferelator 1.0 [2] learns a sparse dynamical model for each $y_i(t)$ as a function of $x(t)$ by assuming that the time evolution in the y's is governed by the ODE:

$$\frac{dy_i(t)}{dt} = \alpha_i y_i + \sum_{j=0}^{P} \beta_{i,j} f_j(x(t)), \qquad i = 1, \ldots, N \tag{5}$$

where $\alpha_i < 0$ is a prior for the degradation rate of y_i,

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_{1,0} & \beta_{1,1} & \cdots & \beta_{1,P} \\ \beta_{2,0} & \beta_{2,1} & \cdots & \beta_{2,P} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N,0} & \beta_{N,1} & \cdots & \beta_{N,P} \end{pmatrix} \tag{6}$$

is a set of parameters to be estimated, $f_0(x(t)) = 1$ (i.e. $\beta_{i,0}$ is a bias term), and each of the other $f_j(x(t))$ is a single variable $x_i(t)$, or the minimum of two variables $\min(x_i(t), x_{i'}(t))$.

Note that the matrix $\boldsymbol{\beta}$ is typically sparse, i.e. most entries are 0, and that the summation term in (5) can fit several types of biologically relevant interactions involving two components (specifically AND, OR, and XOR logical gates) [2].

Least Angle Regression (LARS) [4] is used to efficiently implement an l_1 constraint on $\boldsymbol{\beta}$ [11], which minimize the following objective function, amounting to a least-square estimate based on the ODE (5):

$$\mathcal{E}(\boldsymbol{\beta}) = \sum_{i=1}^{N} \mathcal{E}_i(\boldsymbol{\beta}) \tag{7}$$

where

$$\mathcal{E}_i(\boldsymbol{\beta}) = \sum_{n=1}^{K} \left| \frac{y_i(t_{k+1}) - y_i(t_k)}{t_{k+1} - t_k} + \alpha_i y_i - \sum_{j=0}^{P} \beta_{i,j} f_j(x(t_k)) \right|^2 \tag{8}$$

under an l_1-norm penalty on regression coefficients,

$$\sum_{j=1}^{P} |\beta_{i,j}| \leq s \sum_{j=0}^{P} |\beta_{i,j}^{\text{ols}}| \tag{9}$$

where $\boldsymbol{\beta}^{\text{ols}}$ is the over-fit ordinary least-squares estimate (i.e. the minimizer of (8) with no penalty), and s is a number between 0 and 1 referred to as the shrinkage parameter; setting $s = 1$ corresponds to ordinary least-square regression.

Cross Validation is used to select the value of s that results in models with good generalization, i.e. good predictive performance on new data. Each resulting model is then an ODE describing the time evolution of y_i.

Note that Inferelator 1.0 learns $\boldsymbol{\beta}$ but does not learn the degradation rate α_i, and that minimizing $\mathcal{E}(\boldsymbol{\beta})$ can be done by minimizing the $\mathcal{E}_i(\boldsymbol{\beta})$ sequentially (each such minimization gives a row in $\boldsymbol{\beta}$).

IV. INFERELATOR 2.0

A. Bayesian Approach

To proceed further we note that (5) can be written as a closed equation for the variables y's alone, i.e.

$$\frac{dy_i}{dt} = \sum_{j=0}^{P} \beta_{i,j} f_j(y(t)), \qquad i = 1, \ldots, N \tag{10}$$

and that under this equation $\alpha_i = \beta_{i,i}$.

We assume that the error between the observations $y_i^{\text{obs}}(t_k)$ and the predicted $y_i(t_k)$ can be modeled by a zero-mean multidimensional Gaussian variable. For simplicity, we assume that this error depends on the variable y_i but not on the observation time, and that the errors on the different variables in Y^{obs} are uncorrelated (both assumptions are easy to relax but make notations more cumbersome) i.e. we can write

$$y_i^{\text{obs}}(t_k) = y_i(t_k) + \sqrt{2r_i}\,\eta_i, \tag{11}$$

where η_i are i.i.d. Gaussian variables with mean 0 and variance 1, and $2r_i > 0$ is the variance of the error in $y_i^{\text{obs}}(t_k)$. Under these assumptions, the probability of having observed Y given $\boldsymbol{\beta}$ is

$$p(Y^{\text{obs}}|\boldsymbol{\beta}) = C^{-1} \exp\Big(-\frac{1}{2} \sum_{i=1}^{N} r_i^{-1} \sum_{k=1}^{K} |y_i^{\text{obs}}(t_k) - y_i(t_k)|^2 \Big) \tag{12}$$

where C is a normalization constant. Notice that the dependency of (12) on $\boldsymbol{\beta}$ arises through $y_i(t_k)$, which is the solution of (10) for a given set of $\boldsymbol{\beta}$ evaluated at time t_k.

Using Bayes theorem, (12) can be rewritten to show the probability of $\boldsymbol{\beta}$ given Y^{obs} [5], [7]

$$p(\boldsymbol{\beta}|Y^{\text{obs}}) = \bar{C}^{-1} \exp\Big(-\frac{1}{2} \sum_{i=1}^{N} r_i^{-1} \sum_{k=1}^{K} |y_i^{\text{obs}}(t_k) - y_i(t_k)|^2 \Big) p(\boldsymbol{\beta}) \tag{13}$$

where $\bar{C}$ is another normalization constant (independent of $\boldsymbol{\beta}$) and $p(\boldsymbol{\beta})$ is a prior probability density on $\boldsymbol{\beta}$.

B. Maximum likelihood approach

One natural first step is to find the most likely $\boldsymbol{\beta}$, i.e. the one which maximizes $p(\boldsymbol{\beta}|Y^{\text{obs}})$ [12]. Assuming that $p(\boldsymbol{\beta})$ is uniform in (12), this amounts to minimizing the objective function

$$E(\boldsymbol{\beta}) = \frac{1}{2} \sum_{i=1}^{N} r_i^{-1} \sum_{k=1}^{K} (y_i^{\text{obs}}(t_k) - y_i(t_k))^2 \tag{14}$$

Here $y_i(t_k)$ is the solution of (10) and $E(\boldsymbol{\beta})$ depends on $\boldsymbol{\beta}$ via this solution.

Several efficient methods to minimize (14) (e.g. steepest descent, conjugate gradient, etc.) require the gradient of this function with respect to $\boldsymbol{\beta}$. This gradient is given by

$$\frac{\partial E(\boldsymbol{\beta})}{\partial \beta_{i,j}} = \sum_{i'=1}^{N} r_{i'}^{-1} \sum_{k=1}^{K} (y_{i'}(t_k) - y_{i'}^{\text{obs}}(t_k)) \frac{\partial y_{i'}(t_k)}{\partial \beta_{i,j}} \quad (15)$$

For efficiency we calculate the partial $\frac{\partial y_{i'}(t_k)}{\partial \beta_{i,j}}$ only for the diagonal $i' = i$. Denote by $Z(t_k)$ the $N \times P$ matrix with entries $z_{i,j}(t_k) = \partial y_i(t_k)/\partial \beta_{i,j}$. Then (15) can be expressed as

$$\frac{\partial E(\boldsymbol{\beta})}{\partial \beta_{i,j}} = r_i^{-1} \sum_{k=1}^{K} (y_i(t_k) - y_i^{\text{obs}}(t_k)) z_{i,j}(t_k) \quad (16)$$

and it is easy to see from (10) that Z satisfies

$$\frac{dz_{i,j}}{dt} = f_j(y(t)) + \sum_{j'=0}^{P} \beta_{i,j'} \sum_{i'=1}^{N} \frac{\partial f_{j'}(y(t))}{\partial y_i} \frac{\partial y_{i'}}{\partial \beta_{i,j}}(t) \quad (17)$$

Again calculating the partial $\frac{\partial y_{i'}(t_k)}{\partial \beta_{i,j}}$ only for the diagonal $i' = i$, we can approximate (17) as

$$\frac{dz_{i,j}}{dt} = f_j(y(t)) + \sum_{j'=0}^{P} \beta_{i,j'} \frac{\partial f_{j'}(y(t))}{\partial y_i} z_{i,j}(t) \quad (18)$$

(10) and (18) form a closed system of coupled equations which can be solved numerically using a fast ODE solver to estimate the values of $y_i(t_k)$ and $z_{i,j}(t_k)$ in (16).

If the prior $p(\boldsymbol{\beta})$ is nontrivial, its effect has to be incorporated in the objective function (14).

C. Full posterior sampling via importance sampling MCMC

The major issue with the maximum likelihood approach discussed in IV-B is that the objective function (14) is non-convex in general, i.e. there are many local minima besides the global one. This means that the example minimization procedure discussed in IV-B is likely to get trapped in local minima of $E(\boldsymbol{\beta})$. Thus, it is preferable to return to (13) and proceed with an actual sampling of this posterior distribution for $\boldsymbol{\beta}$ [5], [8].

The natural way to perform this sampling is to use importance sampling Monte-Carlo to generate a sequence $\{\beta_{i,j}^1, \beta_{i,j}^2, \beta_{i,j}^3, \ldots\}$ whose equilibrium density is (13) [8]. A natural possibility is to generate this sequence via

$$\beta_{i,j}^{n+1} = \beta_{i,j}^{n} - h\frac{\partial E(\boldsymbol{\beta}^n)}{\partial \beta_{i,j}} + \sqrt{r_i h}\, \xi_{i,j}^n \quad (19)$$

where $h > 0$ plays the role of updating time step and $\boldsymbol{\xi}^1, \boldsymbol{\xi}^2, \ldots$ are $M \times (P+1)$ matrices where entries $\xi_{i,j}^n$ are independent and identically distributed Gaussian random variables with mean 0 and variance 1. Note that we take advantage of the gradient $\frac{\partial E(\boldsymbol{\beta}^n)}{\partial \beta_{i,j}}$ to sample efficiently high likelihood parameters. (19) is the Euler-Maruyama scheme for the stochastic differential equation [6]

$$d\beta_{i,j}(\tau) = -\frac{\partial E(\boldsymbol{\beta}^n)}{\partial \beta_{i,j}} d\tau + \sqrt{r_i} dW_{i,j}(\tau), \quad (20)$$

Here τ is an artificial time, unrelated to t in (10) and (18), and $W(\tau)$ is an $M \times (P+1)$ matrix whose entries $W_{i,j}(\tau)$ are each a Wiener process (Brownian motion). The equilibrium density of (20) is precisely (13). (19) samples (13) only approximately (to $O(h)$), but it can be incorporated as proposal step in in Metropolis-Hasting Monte-Carlo scheme which samples (13) exactly. This scheme, with a uniform prior was used to generate the results shown in figures 1 and 2. If the prior $p(\boldsymbol{\beta})$ in (13) is nontrivial, it can be incorporated in (19) in the same way as in IV-B.

D. Implementation

We have used *Halobacterium* time series microarray observations from [3], collecting all time intervals, $[t_{k-k'}, t_k]$, with a sampling frequency smaller or equal to 60 minutes, resulting in 335 time intervals. We have then randomly separated these time intervals into two groups: A training set with 224 time intervals, and a testing set with 111 time intervals.

The Inferelator 1.0 ODEs, described in (5), use X as explanatory variables for Y, i.e. use single TFs and TF-TF interactions to explain the rate of change in mRNA levels of gene groupings. In this work we have mapped TFs to their best representative gene grouping (bicluster) to satisfy the closed equation requirement imposed by (10), i.e. a TF is mapped to the bicluster containing it with smallest residual. The resulting data set, D, has 60 gene groupings, representing 60 TFs, measured at the start and end points of 335 time intervals.

We have divided each time interval $[t_{k-k'}, t_k]$ into n equally spaced sub-intervals, where n is a tunable parameter that is currently fixed at five. For each time interval this step resulted in two observed time points (beginning and end) and $n-1$ intermediate time points for which we have no observation. We have used $y_i(t_{k-k'}) = y_i^{obs}(t_{k-k'})$ and $z_{i,j}(t_{k-k'}) = 0$ as initial conditions in (10) and (18), respectively, and numerically solved the two equations to get the values of y_i and $z_{i,j}$ at the first intermediate time point. We have then used these predicted values for y_i and $z_{i,j}$ as initial conditions to predict the next intermediate time point; we have repeated this step until we have reached $y_i(t_k)$ to be used in (14) and $z_{i,j}(t_k)$ to be used in (16). By injecting these intermediate time points we allow predictors to change throughout a time interval and better approximate the coupled, continuous, dynamics of the system.

We have assumed that each y_i has the same observation error $r_i = 1$ in (13), thus it has no effect on the maximum likelihood estimator, i.e. the minimizer of (14), however it may affect rate of convergence. To speed up the method we have used stochastic gradient decent [10] as an estimate for the more exact gradient given by (16). The setup described herein lends itself to an efficient implementation, using iterative matrix multiplication steps. MATLAB code is available from the authors upon request.

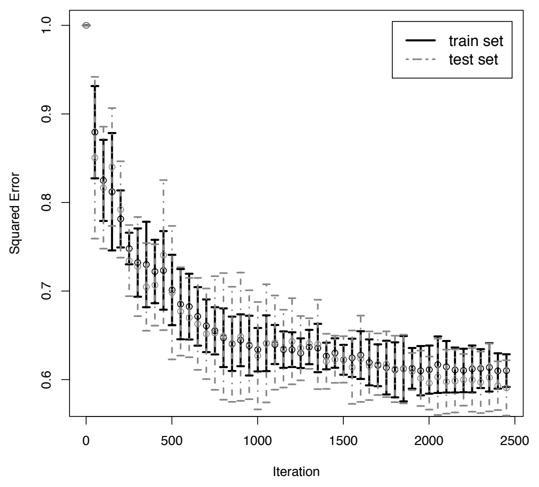

Fig. 1. Train and test set normalized sum-of-squares error as a function of iteration number. For train and test set we have derived normalized errors by dividing $E(\boldsymbol{\beta})$ at iteration $i = 1, \ldots, 2500$ by $E(\boldsymbol{\beta})$ at iteration 1. Error bars are drawn at one standard deviation.

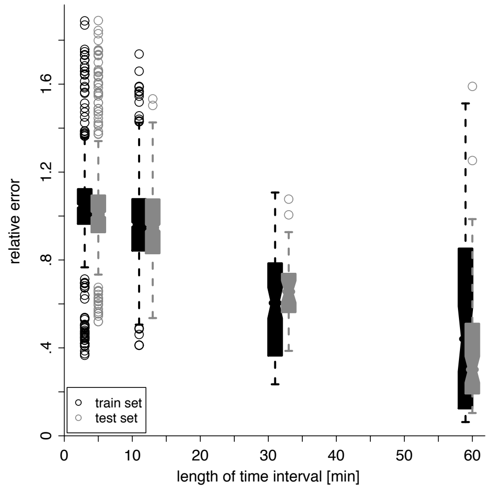

Fig. 2. Relative error as a function of time interval length. We have run Inferelator 2.0 ten times using ten randomly created test and train sets, each time keeping a record of the best learned model, $\boldsymbol{\beta}$. We have then calculated the relative error for each time interval. We have derived relative errors by dividing the sum-of-squares error $(\sum_{i=1}^{N}(y_i^{\text{obs}}(t_k) - y_i(t_k))^2)$ resulting from Inferelator 2.0 by the equivalent sum-of-squares error resulting from Inferelator 1.0. Box plots are shown for time-interval bins of $[1, 5)$, $[10, 20)$, $[20, 50)$ and $[50, 60]$ minutes.

V. Results and Conclusions

Figures 1 and 2, respectively, show that Inferelator 2 improves the overall performance of the RN model without over fitting the train set, and that this improvement becomes more significant for observations corresponding to longer sampling intervals.

Learning biological RNs from systems data is a major challenge in biology today. Two specific sub-challenges that we have focused on in this paper are: 1) finding simple mathematical models capable of describing the more complicated underlying biology, and 2) developing efficient methods capable of effectively exploring the astronomical search space of all possible networks. To address these two sub-challenges we have described a two step approach for network reconstruction.

Our results show that refining our original Inferelator 1.0 derived networks, via Inferelator 2, can significantly boost our previously described ability to model dynamics, specifically improving our ability to model longer time scales. Thus, we propose a two-tiered strategy for network reconstruction where: 1) networks (or ensembles of networks) are constructed using our original approach (which efficiently searches the large space of possible networks) and then 2) refined using our new method (Inferelator 2.0) which balances the longer time-scale dynamic of the network model. We have shown that this framework provides an essential first step towards learning RNs where available data often has longer sampling rates. This is the case in many biological datasets where temporal sampling is constrained by experimental feasibility and cost, and where interesting biology occurs over longer time scales, such as the time scales required to model cell differentiation.

VI. Acknowledgments

The authors gratefully acknowledge the reviewers comments. The work of E. V. -E. was partially supported by NSF grants DMS-0718172 and DMS-0708140, and by ONR grant N00014-08-1-6046. The work of R. B. was partially supported by NSF grant DBI-08207, and DOD grant US-AMRAA W81XWH-04-1-0307.

References

[1] Bonneau, R. Learning biological networks: from modules to dynamics. Nature Chemical Biology **4**, 658–664 (2008).
[2] Bonneau, R., Reiss, D.J., Shannon, P., Facciotti, M., Hood, L., Baliga, N.S. and Thorsson, V. The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo, Genome Biol **7**:R36 (2006).
[3] Bonneau, R., et al. A predictive model for transcriptional control of physiology in a free living cell, Cell **131**:1354–1365 (2007).
[4] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. Least Angle Regression, Annals of Statistics (2003).
[5] Farmer, C. L. Bayesian field theory applied to scattered data interpolation and inverse problems. In Algorithms for Approximation, Editors A. Iske and J. Leveseley, pages 147–166. Springer, 2007.
[6] Gardiner, C. W. Handbook of Stochastic Methods. Springer, Berlin, 1985.
[7] Hastie, T., Tibshirani, R., Friedman, J. The Elements of Statistical Learning, Springer, Berlin, 2001.
[8] Liu, Jun S. Monte Carlo Strategies in Scientific Computing Series. Springer Series in Statistics. Springer, Berlin, 2001.
[9] Reiss, D.J., Baliga, N.S. and Bonneau, R. Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks, BMC Bioinformatics **7**:280 (2006).
[10] Spall C. James. Introduction to Stochastic Search and Optimization. John Wiley and Sons, Hoboken, NJ, 2003.
[11] Tibshirani, R. Regression shrinkage and selection via the lasso, J. Royal. Statist. Soc B. (1996).
[12] Wikle C. K., Berliner M. L. A Bayesian tutorial for data assimilation Physica D: Nonlinear Phenomena, Volume 230, Issues 1-2, June 2007, Pages 1-16.