
A dynamic model of gene regulatory networks based on inertia principle

Florence d'Alché-Buc¹, Pierre-Jean Lahaye¹, Bruno-Edouard Perrin^{1,2},
Liva Ralaivola¹, Todor Vujasinovic², Aurélien Mazurie², and
Samuele Bottani²

¹ Laboratoire d'Informatique de Paris 6, CNRS UMR 7606, 8 rue du capitaine
Scott, 75015 Paris, FRANCE

² Laboratoire de Génétique Moléculaire de la Neurotransmission et des Processus
Dégénératifs, CNRS UMR 7091, Hôpital La Pitié-Salpêtrière, 75013 Paris,
FRANCE

1 Introduction

In molecular biology, functions are produced by a set of macromolecules that interact at different levels. Genes and their products, proteins, participate to regulatory networks that control the response of the cell to external input signals. One of the most important challenge to biologists is undoubtedly to understand the mechanisms that govern this regulation, and to identify among a set of genes which play a regulator role and which are regulated. While the problem used to be approached by a gene to gene approach, this is changed significantly by the development of microarray technology. Expression of thousands of genes of a given organism or a given tissue can now be measured simultaneously on the same chip. This revolution opens a large avenue for research on reconstruction of gene regulatory networks from experimental data.

In this chapter, we claim that both machine learning and modeling of dynamical processes offer a formal and methodological framework to tackle this problem. In our approach, gene regulatory networks are considered as complex, distributed and dynamic systems. The first step consists in the definition of a model of the underlying dynamics. Then, the availability of gene expression kinetics makes it possible to learn parameters of the model. A major advantage of considering the dynamics of the system lies in the fact that the identification step yields both the interaction graph between genes and a simulator for the system. We propose here a linear dynamical model which captures complex behaviours of interactions and a machine learning scheme to identify its parameters. The framework of linear Gaussian state-space models provides a way to take into account noise both in the observations and in the

underlying dynamical process of regulation. We then adapt the well known EM algorithm to our specific model.

We illustrate and evaluate the approach on experimental data that concern the SOS DNA repair network of *Escherichia coli*. While this validation has yet to be extended to other kinds of networks, the results highlight some powerful features of our method such as the introduction of inertia and the management of missing values and noise. The chapter is organized as follows. The problem of reverse engineering of gene regulatory network is described in the second section. Then, in the third section, we introduce an original model based on second order differential equations that handles the presence of inertia. The fourth section is dedicated to learning parameters with a discussion about the merits of two computational intelligence paradigms. Section 5 relates the numerical results we obtained with expression data in *Escherichia coli* within the dynamic bayesian framework. Finally, we present a conclusion and elaborate some future directions in the last section.

2 Elucidating gene regulatory networks from data

Elucidating gene regulatory networks from data can be expressed as a reverse modeling problem. We only observed the expression of genes and we would like to extract from these observations the structural and functional parameters of an appropriate model. Some approaches consider only gene network as static systems and thus, search for the best structure of a boolean network which can be deterministic or probabilistic such as the bayesian networks of [?]. We will not consider them here, focusing on the modeling approaches that allow to simulate the dynamic behavior of the network, once it has been modeled. We need also to highlight the difference between direct modeling and reverse modeling. While direct modeling has been developed during the two last decades especially with the approach of Thomas[?], indirect modeling results from the recent development of high-throughput technologies of DNA chips. Among the existing approaches, one major trend consists in starting from a model of the underlying dynamics of gene interactions and then to apply machine learning tools. The most widely used models are additive [?, ?]. Under this term are gathered models which determine the expression of a gene by using a ponderated sum of all expression levels of the others genes. The simpler among these models is purely linear and determines the expression level E_i of the gene i at instant t by

$$\frac{dE_i^t}{dt} = \sum_j w_{ij} E_j^t + b_i \quad (1)$$

This model does not allow to extract non-linear interactions in the network, but can bring to light the most evident relations. A saturation function can be eventually added to avoid divergences and to make the learning easier.

These latter models have been tested by [?][?] and learned by minimizing a quadratic error criterion.

Bayesian networks can model the expression of each gene as a conditional probability function of the expressions of the other genes. They are therefore well suited for learning from noisy data. Some well-known algorithms for learning Bayesian networks exist [?], and new algorithms for learning very complex models have recently been proposed [?]. [?] gives a very theoretical point of view on the problem of extracting gene interactions from dynamic data. Static Bayesian networks cannot handle temporal information and cycles. Therefore, dynamic Bayesian networks appear to be more adapted to dynamic data. [?] have used dynamic Bayesian networks associated with a discrete model of regulation for modelling regulatory pathways in *E.coli*. An algorithm which identifies interaction networks from dynamic Bayesian networks coupled with a non-parametric regression method has also recently been proposed by [?]. Its main advantage lies on the possibility to reduce noise by using smooth profiles models of kinetics. However the introduction of prior knowledge in this approach does not seem to be easy. In the following, we introduce a new additive model of gene regulatory networks that allow to capture delays and inertia. Then, we will discuss how to attack the problem of its identification.

3 An dynamical model of gene interactions based on an inertia principle

Let us consider two genes A and B , A regulating positively B . If we observe the evolution of their expression levels, we notice that when gene A begins to be expressed at time t_1 , then gene B will be expressed at time $t_1 + \delta t$. On the contrary, if gene A begins to be no more expressed at time t_2 , there is a delay before gene B is itself no more expressed as illustrated in the artificial profiles of gene expression in figure 1. This delay can also be visualized in figure 2 by suppressing time and representing gene B expression according to gene A . We can see an hysteresis that expresses the underlying inertia of the system. In order to reflect the presence of inertia, we propose to model the two genes as coupled dampened oscillators. Such an assumption allows to model complex behaviours of gene expression while keeping the differential equations linear. Indeed, only second-order linear differential equations are needed to translate this inertia. If we consider only the regulation of gene A on gene B , we get the following equations:

$$\frac{d^2 E_B(t)}{dt^2} + 2\lambda_B \omega_B \frac{dE_B(t)}{dt} + \omega_B^2 E_B(t) = w_{BA} E_A(t) \quad (2)$$

Symmetrically, if we now introduce the fact that gene B can also have an action on gene A , we have :

$$\frac{d^2 E_A(t)}{dt^2} + 2\lambda_A \omega_A \frac{dE_A(t)}{dt} + \omega_A^2 E_A(t) = w_{AB} E_B(t) \quad (3)$$

Fig. 1. Gene A up-regulates gene B

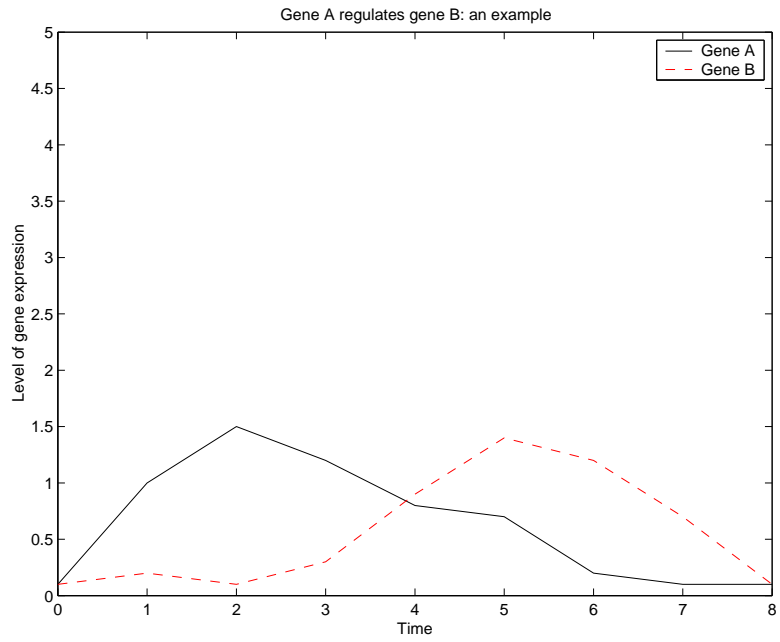
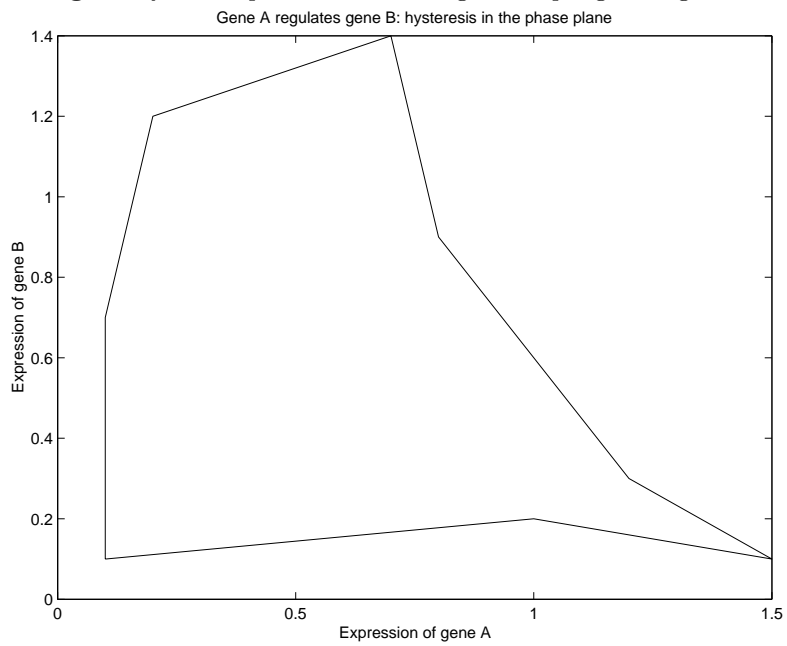


Fig. 2. Hysteresis phenomenon when gene A up-regulates gene B



Of course, it should be emphasized that the model is able to capture an unoscillating behaviour of gene B (resp. A) if ω_B (resp. ω_A) is sufficiently low and λ_B (resp. λ_A) is high enough. We discussed with biologists of a plausible interpretation of the parameters used in these equations:

- λ_B (resp. λ_A) is a coefficient without dimension representing the absorption of the gene B (resp. A). It must be greater than 1 to avoid oscillations. It determines the time necessary to go back to steady state after an excitation, i.e. the time after which no regulated mRNA survives the regulator mRNA. It may be due to the unobserved remanent regulator protein.
- ω_B (resp. ω_A) is the natural frequency of gene B (resp. A), a positive quantity. It defines a characteristic response to an excitation. It takes into account the time delay necessary for transcription and translation to begin.
- w_{BA} is the coupling strength defining the interaction, positive (excitation) or negative (inhibition), exercised by gene A over gene B . $w_{BA} = 0$ indicates that gene A does not influence gene B . It is important to notice that w_{BA} and w_{AB} in the symmetrical equation have no reason to be equal.

The model can be generalized to a network composed of n genes. $E_i(t)$, $i = 1 \dots n$ is the expression level of gene i at time t . Furthermore, we assume that the model is additive e.g. regulatory genes have a linear cumulative effect on a regulated gene. The system is therefore governed by the following system of n second order differential equations:

$$\frac{d^2 E_i(t)}{dt^2} + 2\lambda_i \omega_i \frac{dE_i(t)}{dt} + \omega_i^2 E_i(t) = \sum_j w_{ij} E_j(t), \quad \lambda_i \geq 1. \quad (4)$$

If we introduce $L_i(t)$ as the derivative of expression level for each gene i defined by $L_i(t) = \frac{dE_i(t)}{dt}$, we can now transform the system (4) of n second order differential equations to an equivalent system of $2n$ first order equations:

$$\begin{cases} \frac{dE_i(t)}{dt} = L_i(t) \\ \frac{dL_i(t)}{dt} = -\omega_i^2 E_i(t) - 2\lambda_i \omega_i L_i(t) + \sum_j w_{ij} E_j(t) \end{cases} \quad (5)$$

This system of $2n$ equations can be discretized to be numerically implemented. Assuming that the measurement time unit is lower than the gene evolution characteristic time, we shall consider that continuous derivatives $L_i(t)$ are replaced by: $\frac{\Delta E_i(t)}{\Delta t} = E_i(t+1) - E_i(t)$. We also shall define the vector \mathbf{x}_t as the state vector defined by :

$$\mathbf{x}_t = \left(E_1(t), \dots, E_n(t), \frac{\Delta E_1(t)}{\Delta t}, \dots, \frac{\Delta E_n(t)}{\Delta t} \right)' \quad (6)$$

. Using these notations, the evolution of the network of n genes can therefore be described by the following equation

$$\mathbf{x}_{t+1} = A \cdot \mathbf{x}_t \quad (7)$$

with matrix

$$A = \begin{bmatrix} I & I \\ W - \Omega^2 & I - 2\Omega\Lambda \end{bmatrix} \quad (8)$$

where I is the identity matrix of size $n \times n$, $W = (w_{ij})_{1 \leq i, j \leq n}$, $\Omega = \text{diag}(\omega_1, \dots, \omega_n)$ and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ ³.

The linearity of the system provides two advantages:

- stability can easily be analysed⁴
- the identification process will be simpler than for a non linear system.

4 Learning parameters of the inertial model

4.1 Machine learning methodology

Machine learning aims at inferring mathematical models from examples in order to solve complex tasks such as pattern recognition and data-mining. In this framework, a mathematical model refers to a probability law, a set of rules or a function chosen in a given family. The learning process is considered as successful if the learned system can deal correctly with new data that come from the same parent probability law. This ability to behave correctly in front of new data is called the *generalization property*. Since 1985, machine learning have developed both theoretical and empirical tools to ensure that learned models generalize well. In the following we first consider the whole learning process through a general methodology that includes 3 main steps: representation, optimization and validation. At least the two first steps are somehow interdependent and require prior knowledge to be processed. The combined use of data and knowledge is characteristic of artificial intelligence and machine inference. We think that such an approach is especially well-suited to a modeling process and we show it in the context of the reverse modeling problem. Hence, reverse engineering of gene regulatory networks can be formalized as follows: Let us note $S = \mathbf{x}_1, \dots, \mathbf{x}_T$, the observed time-series of mRNA concentrations of the n genes. We assume that H is a given family of functions $h_\theta(\mathbf{x}_t) = \mathbf{x}_{t+1}$. θ is the set of parameters of h . We also refer to K as the set of prior available knowledge. We aim at defining a learning algorithm a such that:

$\hat{\theta} = a(S, H, K)$ such that the generalization ability of $h_{\hat{\theta}}$ should be maximal.

Defining the algorithm a requires first to decide how to represent the data, which model to implement. this is the *representation step*. It also needs to

³ In this chapter $\text{diag}(d_1, \dots, d_n)$ is the diagonal matrix of size $n \times n$ whose diagonal elements are d_1, \dots, d_n and the others zero; for a square matrix M , $\text{diag}(M)$ is the diagonal matrix whose diagonal elements are the diagonal elements of M .

⁴ Let us recall that the linear system described by equation 7 is stable if and only if the eigenvalues of the matrix A are of magnitude equal or inferior to 1.

choose according what criterion the parameters θ have to be optimized and how this can be performed: it is the *optimization step*. Lastly, the results have to be evaluated and validated either in an automatic way or with the help of the biologist. Let us detail each of these questions to be addressed:

- the **Representation** question : we must decide how to encode the available data and which class of models we need to fit them. In our problem, the raw data are kinetics of mRNA concentration for each gene. We have to decide if we work directly on these variables with continuous values or if we discretize them. We can also extract some features for instance with signal transformations and use them as new variables. In this case, preprocessing the data makes the role of the model easier. Choosing a family of models requires to take into account two important properties : one one hand, the class of models has to be sufficiently expressive to represent accurately the system to be modeled and on other hand, each model needs to be flexible in order to get its complexity controlled. For function approximation and time-serie modeling, the control of the norm of the parameters allows usually to keep the model simple.
- the **Optimization** question : learning has to be set as an optimization problem where objective function is clearly defined. The objective function to be optimized must reflect the goal of the learning process which is to get good generalization properties. Several theories from regularization one [?] to those of large deviations theorems [?] prone to define a cost function that makes the balance between empirical error and sparsity of the model. Your model is thus required to fit the available data while keeping simple: this has been proved to be successful both theoretically and empirically for years of pattern recognition and machine learning research. When the optimization problem is established, an optimization algorithm is developed to determine the parameters of the given model. According to the nature of the problem to be solved, mathematical programming, gradient descent or Expectation-Maximization algorithms and evolutionnary algorithms can then be used. It should be emphasized that the relevance of the learning results depends not only on the appropriatness of the cost function but also of the convergence properties of the optimization algorithm. Some algorithms are proven to fall into local minima of the cost function and this has to be taken into account when exploiting the results.
- the **Validation** question: as learning is performed using a finite sample of data, it must be put into question. Once of criterion is defined to evaluate the quality of the result, robust methods of statistical estimation such as bootstrap or cross-validation can be used. Within the context of time-series analysis and dynamic modeling, the classic criteria which can be estimated are the ability to make predictions, for instance one of k steps ahead.

Yet there is another kind of validation: the validation by the domain expert who is here the biologist. The learning process provides an hypothetic

model of gene interactions that the biologist must check either by comparing the obtained model with existing true model or by defining some experiments that will prove or refute the hypothetic model. In the numerical results we present, we used the two first kinds of validation.

After this introduction to machine learning principles, we now describe how we have exploited biological knowledge to solve the representation and the optimization questions for the reverse modeling problem.

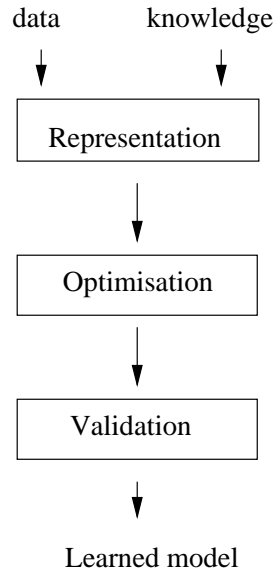


Fig. 3. Machine learning methodology

4.2 Solving the representation issue using the inertial model

In the following, we consider that the data encoding is very simple : we keep kinetics of gene expression with continuous values and avoid to discretize them in order to add quantization noise. The choice of the models family is now reduced to the implementation of the inertial model in a framework that favours learning.

Deterministic implementation as a recurrent artificial neural network

The linear dynamic model that we proposed belongs to the family of linear recurrent artificial neural networks with a specificity: the equations we have

derived from the differential equations describe a distributed system composed of two kind of units. We have units that can be associated with genes which correspond to the E_i variables and another kind of units, the latent units which correspond to the L_i variables. Information propagates from t to $t + 1$. So the network can be described as recurrent since output at time $t + 1$ become inputs to compute the state of the network at time $t + 2$ and so on. Within this framework, learning could be performed using backpropagation through time algorithm (BPT) with a possible non linearization of the units to improve the learning process. However we identified one major drawback for this approach: it does not handle noise specifically and moreover, it does not address the missing variables problem which is a frequent case in biological settings. Therefore, another point of view is to consider that observed data are produced by a latent linear model corrupted by intrinsic noise.

4.3 Probabilistic implementation as a linear Gaussian state-space model

The idea is to implement the inertial model as a linear Gaussian state-space model. First we assume that the true process modeled by the equation 7 is hidden and only accessible by the observations that it produces. Second we propose to take into account two sources of noise: the intrinsic noise which is known to be present in genetic expression [?] and measurement noise which is due to the acquisition technologies.

In this work, we have made the simplest assumptions about noises and have focused on the gaussian hypothesis, either for intrinsic or measurement noise.

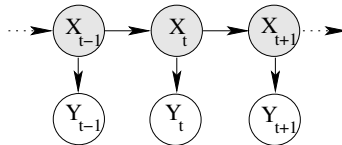


Fig. 4. Linear dynamical system. Dark nodes are hidden. Bright nodes are observed.

Using the linear dynamical system formalism (cf. Figure 4), these assumptions lead to the following model:

$$\begin{cases} \mathbf{x}_{t+1} = A.\mathbf{x}_t + \mathbf{u} \\ \mathbf{y}_t = C.\mathbf{x}_t + \boldsymbol{\mu}_{obs} + \mathbf{v} \end{cases} \quad (9)$$

X_t is the hidden state of the gene network at instant t (cf. Equation 6), while $vec(y)_t$ is the observed state of the network, composed of all observations of gene expression levels. A is the transition matrix of Equation 8 and C the projection matrix $[I \quad \mathbf{0}_{n,n}]$ of size $n \times 2n$, $\mathbf{0}_{m,n}$ being the zero matrix of size

$m \times n$, and $\boldsymbol{\mu}_{obs}$ a measurement adjustment vector. Elements \mathbf{u} and \mathbf{v} are independent and identically distributed (i.i.d.) realizations of two Gaussian random variables with zero mean and variances σ_x^2 et σ_{obs}^2 . \mathbf{u} and \mathbf{v} express the fact that both biological and measurement phenomena are stochastic. Variables X_t are usually said to be *hidden* because they only are accessible indirectly through observation of Y_t . We make the hypothesis that X_1 follows a Gaussian law of mean $\boldsymbol{\mu}_i$ and variance σ_i^2 .

The proposed model (9) can be also be described explicitly using conditional probabilities and can be seen as a *dynamic Bayesian network* (DBN) with hidden nodes (cf. Fig.??)

$$\begin{cases} p(\mathbf{x}_{t+1}|\mathbf{x}_t) = N(\mathbf{0}, \sigma^2 I) \\ p(\mathbf{y}_t|\mathbf{x}_t) = N(\boldsymbol{\mu}_{obs}, \sigma_{obs}^2) \end{cases} \quad (10)$$

Dynamical Bayesian networks describe relations of conditional dependencies on time-dependent variables. The proposed model also belongs to the family of *Kalman filter models* which was proposed in the late sixties by Kalman to process signal filtering and smoothing. In the following, we will refer to our model as the Inertial Dynamic Bayesian Network (IDBN). "

4.4 The optimization scheme

In the context of learning from a unique time-serie⁵, we can define a cost function that should be minimized to get good generalization abilities. Within the probabilistic framework, a natural criterion is the log likelihood of the data, $L(\theta; \mathbf{y}_1, \dots, \mathbf{y}_T)$:

$$L(\theta; \mathbf{y}_1, \dots, \mathbf{y}_T) = \log p(\mathbf{y}_{1:T}|\theta) \quad (11)$$

In order to get a criterion over all the possible time-series, we can now derive from this definition the functional risk $R(\theta)$ as the expectation of minus the log-likelihood calculated over all time-series $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_T)$:

$$R(\theta) = - \int L(\theta; \mathbf{y}_1, \dots, \mathbf{y}_T) p_{true}(\mathbf{y}_{1:T}) d\mathbf{y}_{1:T} \quad (12)$$

If our goal is to optimize generalization ability of the model, we need to minimize the functional risk $R(\theta)$. However, as the true density p_{true} is not known (it is what we are looking for), this quantity cannot be calculated. However both statistical learning [?] and regularization theories [?] provide us with a number of fundamental results that prone to replace this criterion by the combination of two criteria : one is based on the empirical estimation of $R(\theta)$, the other controls the complexity of the model. The underlying idea beyond this is that as far as the model fits correctly the data while it keeps simple, there will be no overfitting and the obtained model will be enough general to deal with other time-series generated from the same distribution. The empirical cost function will be here the log-likelihood of the observed data given

in equation???. Defining a criterion that reflects the complexity of the model appears to be a more difficult point. However if we consider the underlying graph of interactions between genes which is represented by the W matrix, we can assume that a simple model for genetic interactions is a sparse model with few connections between genes. From a biological point of view this assumption corresponds to the observed fact that each gene interact with very few other genes (statistics of input and output degrees for instance for yeast can be found in[?]). In the optimization scheme,controlling the complexity of the model means to encourage sparsity of the W matrix during learning and this can be operated by minimizing the norm of matrix W $\|W\|$. Another equivalent way to interpret this combination of criteria is to consider the parcimony assumption as a prior in the framework of bayesian inference. Instead of minimizing only $-\log P(\mathbf{y}_{1:T}|\theta)$ we will minimize also $-\log[p(\mathbf{y}_{1:T}|W, \theta').p(W)]$. Assuming that the prior probability on W is Gaussian using $L-1norm$, this leads to the following criterion :

$$R(\theta; \mathbf{y}_1, \dots, \mathbf{y}_T) = -\log p(\mathbf{y}_{1:T}|W, \theta') - \log p(W) \quad (13)$$

with $p(W) \propto \exp(-\alpha \sum_{ij} |w_{ij}|)$.

α is called the regularization hyperparameter: it weights the effect of the parcimony constraint applied to the parameters.

4.5 The optimization algorithm

Parameters can be learned using a generalization of *Expectation-Maximization*(EM) algorithm [?, ?] and wich has been thoroughly described for linear Gaussian processes in [?] in a unifying review. EM algorithm, which allows to handle the hidden variables \mathbf{x}_t , is an iterative process that uses the following two steps :

The standard *Expectation* phase implements the *filter* and *smoother* processes, as described in Rosti01. This phase is summarized in Table 1. It allows to determine directly the most probable states \mathbf{x}_t given $\mathbf{y}_{1:T}$. The *Maximization* phase is different from the usual one, because of the specific nature of matrix A .

The auxiliary function $Q(\theta, \theta^{(k)})$, parametrized by $\theta^{(k)}$, the current estimation of parameters θ at step k , is defined as the expectation (operator $E[\cdot]$) of the penalized log-likelihood Ormoneit98 with respect to $\mathbf{y}_{1:T}$: $Q(\theta, \theta^{(k)}) = E[\mathcal{L}^{pen}(\theta)|\mathbf{y}_{1:T}, \theta^{(k)}]$.

The M phase used for our algorithm determines $\theta^{(k+1)}$ by making a gradient step in the direction $\nabla_{\theta} Q(\theta, \theta^{(k)})$ from $\theta^{(k)}$, parameters estimated after k EM iterations:

$$\theta^{(k+1)} = \theta^{(k)} + \eta \nabla_{\theta} Q(\theta, \theta^{(k)}) \quad , \quad \eta > 0 \quad (14)$$

More precisely, the M phase consists in the following computations:

- $\boldsymbol{\mu}_i^{(k+1)} = (1 - \eta)\boldsymbol{\mu}_i^{(k)} + \eta\hat{\mathbf{x}}(1)$
- $\sigma_i^{2(k+1)} = (1 - \eta)\sigma_i^{2(k)} + \eta \left[\frac{1}{2n}(\hat{R}(1) - \boldsymbol{\mu}_i^{(k)}\boldsymbol{\mu}_i^{(k)'}) \right]$
- $\boldsymbol{\mu}_{obs}^{(k+1)} = (1 - \eta)\boldsymbol{\mu}_{obs}^{(k)} + \eta \left[\frac{1}{T} \sum_{t=1}^T (\mathbf{y}_t - C\mathbf{x}_t) \right]$
- $\sigma_{obs}^{2(k+1)} = (1 - \eta)\sigma_{obs}^{2(k)} + \eta \left[\frac{1}{nT} \left(\sum_{t=1}^T (Y_t Y_t' - C\hat{\mathbf{x}}(t)\mathbf{y}_t' - \boldsymbol{\mu}_{obs}^{(k)}\mathbf{y}_t') \right) \right]$
- $\sigma_x^{2(k+1)} = (1 - \eta)\sigma_x^{2(k)} + \eta \left[\frac{1}{n(T-1)} \left(\sum_{t=2}^T (\hat{R}(t) - A^{(k)}\hat{R}^{t-1}(t)) \right) \right]$

For updating parameters $W^{(k)}$, $\Omega^{(k)}$ and $\Lambda^{(k)}$, it is useful to consider the G_k matrix defined by:

$$G_k = \frac{1}{\sigma_x^{2(k)}} \sum_{t=2}^T \left[\hat{R}^{t-1}(t) - A^{(k)}\hat{R}(t-1) \right] \quad (15)$$

and more precisely the matrices of size $n \times n$ $G_k^{\ell\ell}$ and $G_k^{\ell r}$ which denote respectively the G_k lower left and lower right submatrices. Parameters $W^{(k+1)}$, $\Omega^{(k+1)}$ and $\Lambda^{(k+1)}$ are computed according to:

$$\begin{aligned} W^{(k+1)} &= W^{(k)} + \eta(G_k^{\ell\ell} - \lambda \nabla_W ||W||) \\ \Omega^{(k+1)} &= \Omega^{(k)} - 2\eta \text{diag} (G_k^{\ell\ell} \Omega^{(k)} + G_k^{\ell r} \Lambda^{(k)}) \\ \Lambda^{(k+1)} &= \Lambda^{(k)} - \eta \text{diag} (G_k^{\ell r} \Omega^{(k)}) \end{aligned}$$

Once the $A^{(k+1)}$ matrix is computed according to Equation (8), it is possible to proceed to a new E phase. At each EM step, the penalized likelihood increases, until a local maximum is reached.

Filter	Smoother
$\mathbf{x}^{t-1}(t) = A\mathbf{x}^{t-1}(t-1)$ $\Sigma^{t-1}(t) = A\Sigma^{t-1}(t-1)A' + \sigma_s^2 I$ $\Sigma_e(t) = C\Sigma^{t-1}(t)C' + \sigma_{obs}^2 I$ $K_t = \Sigma^{t-1}(t)C'\Sigma_e^{-1}(t)$ $\mathbf{e}_t = \mathbf{y}_t - C\mathbf{x}^{t-1}(t) - \boldsymbol{\mu}_{obs}$	$J_{t-1} = \Sigma^{t-1}(t-1)A'(\Sigma^{t-1}(t-1))^{-1}$ $\hat{\mathbf{x}}(t-1) = \mathbf{x}^{t-1}(t-1) + J_{t-1}(\hat{\mathbf{x}}(t) - \mathbf{x}^{t-1}(t))$ $\hat{\Sigma}(t-1) = \Sigma^{t-1}(t-1) + J_{t-1}(\hat{\Sigma}(t) - \Sigma^{t-1}(t))J_{t-1}'$ $\hat{\Sigma}^{t-1}(t) = \hat{\Sigma}(t)J_{t-1}'$
$\mathbf{x}^t(t) = \mathbf{x}^{t-1}(t) + K_t\mathbf{e}_t$ $\Sigma^t(t) = \Sigma^{t-1}(t) - K_t C \Sigma^{t-1}(t)$	$\hat{R}(t) = \hat{\Sigma}(t) + \hat{\mathbf{x}}(t)\hat{\mathbf{x}}'(t)$ $\hat{R}^{t-1}(t) = \Sigma^{t-1}(t) + \hat{\mathbf{x}}(t)\hat{\mathbf{x}}'(t-1)$

Table 1. Filter and smoother equations. $\mathbf{x}^0(1) = \boldsymbol{\mu}_i$, $\Sigma_1^0 = \sigma_i^2 I$, $\hat{\mathbf{x}}(T) = \mathbf{x}^T(T)$ et $\hat{\Sigma}(T) = \Sigma^T(T)$. $\hat{R}(t)$ and $\hat{R}^{t-1}(t)$ are used for determining the model parameters.

Discussion

Let us notice an important feature of the IDBN model : if some genes are to play a decisive role in the network, but if no expression data is available for them, it is however possible to include them by modifying some of the model features. These genes will be called *missing variables*. The proposed framework is adapted to handle them: the model is divided in two different spaces, the *hidden space* of \mathbf{x}_t and the *observation space* of \mathbf{y}_t ; there is only a projection matrix C to go from hidden state to observed state. Let us study a network composed of $n+h$ units, with available data for n genes, but the others h genes being unmeasured. Vectors \mathbf{y}_t of the observation space do not change and are always of size n . On the contrary, vectors \mathbf{x}_t are now of size $2(n+h)$, being composed of all genes, including the missing ones. A is now a $2(n+h) \times 2(n+h)$ matrix, and the projection matrix C is $[I \quad \mathbf{0}_{n,h} \quad \mathbf{0}_{n,n+h}]$ of size $n \times 2(n+h)$. Incorporating missing variables in our framework is therefore very easy, and it does not change anything to the use of the EM learning algorithm.

5 Experiments

5.1 Experimental data sets

In order to validate our approach, we considered the *S.O.S. DNA Repair* network of the *Escherichia coli* bacterium. The goal of is to learn the parameters of the inertial model in order to fit the available experimental data. No prior knowledge on the structure of the network is used except the average connectivity degree which is supposed to be given. Validation is first realized by measuring how the identified model is able to do prediction one or K step ahead and by comparing between the learned model and the true network.

5.2 Experimental data sets

The well-known regulation network, *S.O.S. DNA Repair*, is responsible for repairing the DNA after a damage. The whole system is composed of about 30 genes regulated at the transcriptional level. Usually, when no DNA damage occurs, a master transcription factor *LexA* binds sites in the promoter regions of these genes, repressing all genes of the network. One of the *S.O.S.* proteins, *RecA*, acts as a sensor of DNA damage: by binding to single-stranded DNA, it becomes activated and mediates *LexA* destruction. The drop in *LexA* levels causes the de-repression (i.e. activation) of *S.O.S.* genes. Once damage has been repaired or bypassed, the level of activated *RecA* drops, *LexA* accumulates and represses the *S.O.S.* genes, and cells return to their initial state.

Experimental data have been made available by Uri Alon (they are downloadable on its homepage ⁶). Data are expression kinetics of the main 8 genes

⁶ <http://www.weizmann.ac.il/mcb/UriAlon/>

Fig. 5. *S.O.S. DNA Repair* network. Activations are represented by arrows (\rightarrow) and inhibitions by T (\dashv). Genes initials are in lower cases, proteins in capital letters.

of the *S.O.S. DNA Repair* network of *E.coli*. The measurement technology is based on the property of the GFPs (green fluorescent proteins). Alon have developed a system for obtaining very precise kinetics [?]. Measurements are done after irradiation of the cells at the initial time with UV light. Four experiments are done for various light intensities (Exp. 1&2 : $5 Jm^{-2}$, Exp. 3&4 : $20 Jm^{-2}$). Each experiment is composed of 50 instants evenly spaced by 6 minutes intervals, and 8 genes are monitored: *uvrD*, *lexA*, *umuD*, *recA*, *uvrA*, *uvrY*, *ruvA* and *polB*.

We first explain why the data are useful for our purpose and can be incorporated in our model. Alon monitor each *S.O.S.* gene separately, adding its promoter to a *gfp* sequence in a plasmid and incorporating the plasmids in irradiated *E.coli*. The quantity of present GFP, which is indirectly measured by the amount of fluorescence, is therefore proportional to the quantity of the corresponding *S.O.S.* protein. The first hypothesis made by Alon is that the GFP protein is very stable during the experiments: it is justified when comparing the typical GFP stability to the experiments length (300 minutes). By measuring the time-course of fluorescence intensity, Alon and coworkers have therefore access to the instantaneous protein production rate, since no protein degradation occurs during the experiments.

By making the standard hypothesis that the protein production rate is proportional to the corresponding mRNA production rate, they consider that the derivatives of the fluorescence amounts are proportional to the promoter activity of the genes. It is important to notice that this hypothesis is not so hard as usual in this case, because all proteins are the same (GFP protein). One more difficult point is why Alon et al. divide also by the OD, but we will not discuss this point here. Figure 3 (b) of [?] indicates these promoter activities.

To use these data in our model, we have to make a strong hypothesis on the stability of the mRNAs: we consider that mRNA molecules are degraded immediately after their production. Actually mRNA persistence depends on the nucleotide sequence; all mRNAs having the same sequence here because of the experimental technique, they all have the same persistence. It is hence sufficient to make the hypothesis that the turnover of the GFP mRNAs is very fast to ensure that there is a high unstability of the mRNAs. We therefore consider that the instantaneous promoter activity of each gene is also proportional to the present quantity of corresponding mRNA.

We hence are able to consider that the data provided by Alon directly indicate the observed mRNA quantities (also called expression levels) corresponding to each *S.O.S.* gene. The downloaded data consists in four 8×50 matrices corresponding to the four stresses. The t^{th} column of a given matrix

is considered to be the observation vector \mathbf{y}_t , and the entire matrix will be considered as the time course $\{\mathbf{y}_1, \dots, \mathbf{y}_{50}\}$.

5.3 Learning experiments

We have proceeded to several learning experiments on the data provided by Alon with the Inertial Dynamical Bayesian Network (IDBN). The influence of the regularization parameter α is important and has been carefully studied. For each data set, we have made successive learnings with $\alpha = 0, 1, 5, 10, 50, 100, 500, 1000$ and 5000 . In each case, we have introduced 0 or 1 missing variable. The EM algorithm stops after 100 iterations, which is sufficient, since not discussed here previous experiments have shown that after 80 iterations, more than 95% of the parameters will not change of more than 1%, which does not change significantly the learned model. Parameters are initialized as follows: each coefficient of W is randomly initialized between 0.01 and 0.01, Λ and Ω are chosen from the assumption that unoscillatory behaviour is observed ($\lambda_i \simeq 1$ and $\omega_i \simeq 0 \forall i$), each coefficient of μ_i and μ_{obs} between 0 and 1, and finally σ_i^2 , σ_x^2 and σ_{obs}^2 between 0 and 0.1. The main crucial point seems to be the initialization of W , Λ and Ω . 50 different learnings under each condition have hence been made: this *multiple random starting points* technique is widely used to handle the problem of likelihood local maxima.

5.4 Capturing the network dynamics

Capturing the dynamics characteristics

The proposed method is able to capture the network dynamics. Figure 6 allows to compare the real profiles of the 8 genes relative to the second experiment and their simulated profiles corresponding to the learned model for $\lambda = 100$. It is important to notice that these simulated profiles are *mean profiles*, since the variances associated to the model (σ_i^2 , σ_x^2 , σ_{obs}^2) are not taken into account in this simulation. The behaviour of the network according to these elements will be discussed later. The variations of the most expressed genes of the network (*recA*, *lexA*, *uvrA*, and to a lesser extent *umuD* and *uvrD*) are finely modelled. One can notice that the respective maxima of these genes are reached at different instants whose succession is essential to explain the functioning of the *S.O.S.* response system: this succession is respected in the learned model.

No noteworthy dose effect has been observed when comparing results of the different experimental conditions. It only appears that gene expressions levels are higher when the irradiation intensity increases, but the respective maxima and variations do not change between the four experiments, indicating that the *S.O.S.* system has the same type of answer for both experiment intensities.

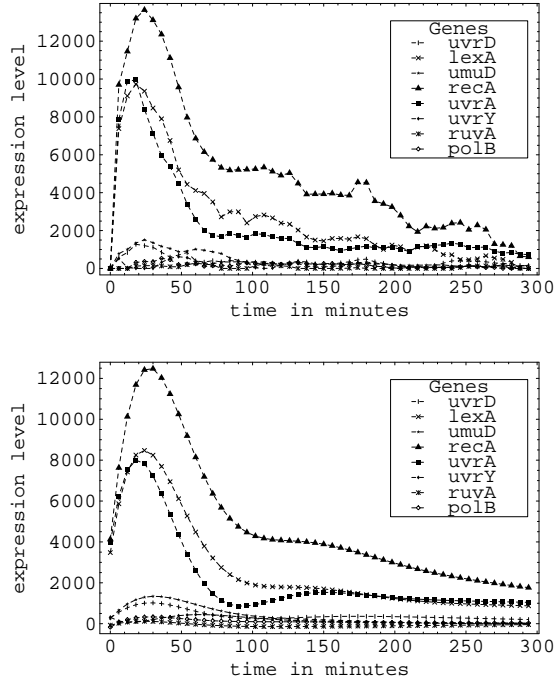


Fig. 6. Top: expression profiles of the 8 genes corresponding to the second second data set. Bottom: learned profiles from this data set for $\lambda = 100$. The vertical scale has no absolute meaning, since we took the $(dGFP/dt)/OD$ values of Ronen02, because they are proportional to the expression levels of the genes as explained previously. It is important to notice that these simulated profiles are *mean profiles*, since the variances associated to the model are not taken into account in this simulation.

Capturing the stochastic phenomena

One of the characteristics of our model is its ability to represent stochastic phenomena: one could wonder if the stochastic learned parameters are compatible with the real data. On Figure 7, we have represented five simulated profiles of *lexA* and *recA* using the model learned from the second data set with $\lambda = 50$.

For simulating such profiles, we use Equations 9. At each time point, \mathbf{u} and \mathbf{v} are randomly chosen according to their Gaussian distribution with variances σ_x^2 and σ_{obs}^2 . The system is also randomly initialized according to $\mathbf{x}_1 \sim \mathcal{N}(\mu_i, \sigma_i^2 I)$. We also have represented the mean profiles as in Figure 6 setting the variances to 0, and the measured profiles of the second data set. One can notice that the measured profile and the mean simulated profile do not merge: nevertheless, the measured profile is located in the envelope of

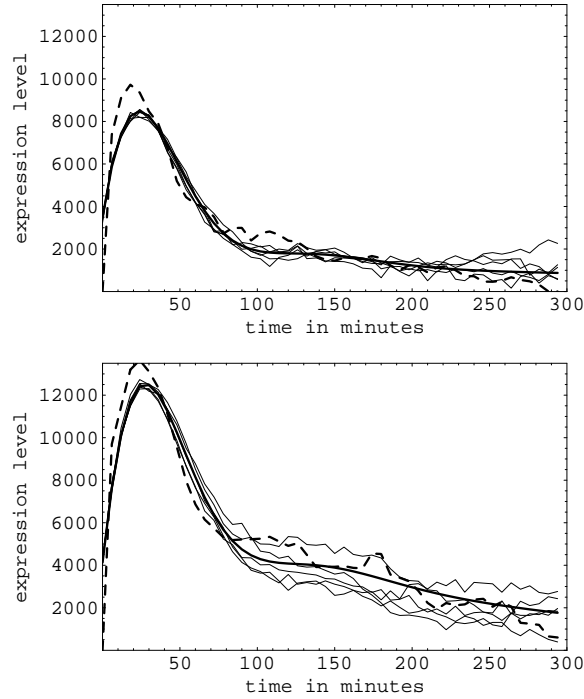


Fig. 7. Profile of *lexA* (top) and *recA* (bottom) under the same conditions as in Figure 6. The thick dotted line is the measured profile given in the data set; the thick continuous line is the mean learned profile already represented on Figure 6; thin continuous lines are simulated profiles, taking into account the learned model variances.

simulated stochastic profiles (at least during the decrease). This shows that the learned variances are compatible with real variances.

For space commodity, we do not have plotted here such simulated profiles for all the genes under others conditions. Nevertheless experiments allow us to conclude that model variances associated with other learned model parameters are in accordance with the observed profiles. Our learning technique associated with the model choice is hence able to capture the mean dynamics of the network, so as to take into account stochastic phenomena, either due to measurement noise or to regulation itself.

5.5 Structure extraction

Principle

It has been shown previously that our model is based on the assumption⁷ that co-regulation is an additive phenomenon. While it is not always true, this assumption allows to associate a biological meaning with each w_{ij} parameter according to this simple rule :

- $w_{ij} > 0$: gene j activates gene i
- $w_{ij} < 0$: gene j inhibits gene i
- $w_{ij} = 0$: gene j does not regulate gene i

After each learning, a W matrix is identified. According to the previous assumption, it should directly indicate the regulations in the network. Because of the presence of local maxima of the likelihood function, the identified W is not always the same at each numerical experiment. However we show that the regularization process can alleviate the problem.

Impact of the regularization process

The regularization technique is based on the simple idea that gene networks are known to be usually sparse: most of the genes have few regulators, and in turn regulate few genes. This regularization is very standard in the machine learning framework and is known to favour such sparse networks, which is biologically motivated in our case. One could object that the regularization term does not encourage sparseness, but simply low norm for W , so as a matrix with many weak connections can be as favorable as one with few strong ones. In fact, it does not appear to behave like that. Of course, each w_{ij} coefficient will decrease as the regularization parameter α increases, but some coefficients decrease more slowly than others.

In order to illustrate this phenomenon, which proves the efficiency of our penalizing term, we have made some statistics on our experimental results. For each data set, we have studied the W matrix learned for three different regularization parameters $\alpha = \{0, 100, 1000\}$ with no added hidden variable. We have computed the mean and standard deviation of all coefficients, and plotted them on Figure 8. Coefficients are ordered by mean. The plotted figure shows the results obtained for the first data set, but the others are similar.

It seems obvious that the curve flattens as the regularization parameter α increases. This results proves that our regularization term encourages a real sparseness, and not only low coefficients. Of course, one can notice that all means decrease with α , but some coefficients remain strictly non zero: this is the main goal of our regularization. Moreover, we also can notice that standard

⁷ This assumption is common to all the models based on differential equations which have been proposed so far in the literature

deviations decrease with α : it should suggest us that a good regularization decreases the search space dimension, and in consequence the number of local maxima of the likelihood function. Nevertheless, when α is too high, all coefficients tend to zero, so that all interesting informations about regulations vanish.

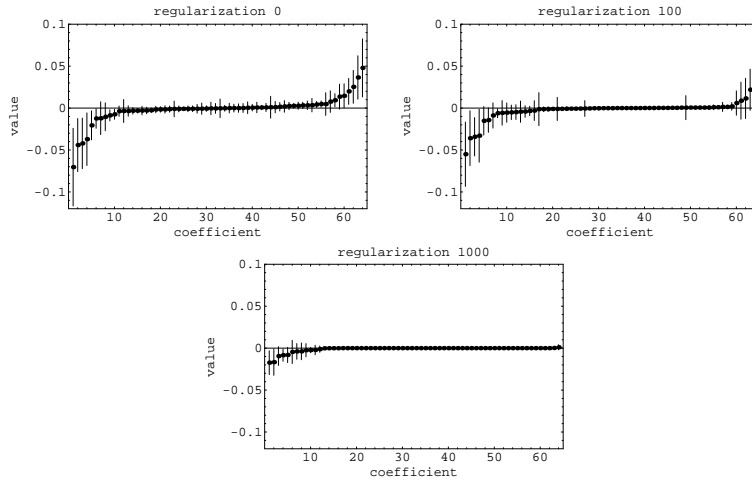


Fig. 8. W coefficients learned from the first data set, using respectively regularization 0, 100 and 1000. Coefficients are ordered by their mean on the 50 learnings. Error bars indicate their standard deviation.

There are several ways to select or learn α value. If numerous independent time series were available, it would be possible to apply cross-validation to select its right value. Another method consists in applying a full Bayesian approach by assuming a prior on α and incorporating in the algorithm a learning stage for α . In our framework, the regularization parameter can be chosen according to the average degree k of the genetic graphs. The average degree of a graph is given by $k = \frac{2M}{N}$ where N is the number of nodes, and M the number of arcs. It indicates the average number of arcs bound to a node. Some data are available about the topological structure of transcriptional networks: ShenOrr02 give 577 interactions for 116 transcription factors in *E.coli* using the RegulonDB database, which leads to $k = 9.95$, while Guelzim02 propose a yeast transcriptional network composed of 491 genes and 909 transcriptional interactions giving $k = 3.70$. Unfortunately, the *S.O.S.* network has a very particular *star-like* topology, and is much smaller. We hence cannot use these average degrees for finding the best regularization. For this experiment, we hence use the average degree of the actual network. Figure 5 gives $k = 2.25$ for transcriptional regulations in the *S.O.S.* network considering that there are 9 transcriptional regulations between the 8 genes. As we find $k = 4.25$ for $\alpha = 0$, $k = 2.25$ for $\alpha = 100$, and $k = 1$ for $\alpha = 1000$ (the way to obtain the

number of identified regulations is discussed further), we will consider that the optimal regularization parameter is 100. This regularization allows to make a compromise between favouring the sparseness of W and keeping information about regulations.

Identifying regulations

For each data set, with $\alpha = 100$, 50 learnings starting from random starting points are done as explained previously. The learned values of each parameter w_{ij} are distributed with mean μ_{ij} and variance σ_{ij}^2 . The mean and variance of the means of all 64 coefficients named μ and σ^2 can also be computed. Coefficients are then discretized into four classes according to their mean and standard deviation:

- class [+]: $\mu_{ij} > \mu + \sigma$ and $\sigma_{ij} < |\mu_{ij}|$
- class [-]: $\mu_{ij} < \mu - \sigma$ and $\sigma_{ij} < |\mu_{ij}|$
- class [0]: $|\mu_{ij}| < \sigma$ and $\sigma_{ij} < \sigma$
- class [X]: others coefficients

Classes are built to represent respectively probable activations, probable inhibitions, probable absences of regulation, and probable presences of unknown regulations. The total number of regulations in the network is obtained adding the number of coefficients in the classes [+], [-] and [X].

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & - & + & + & - & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & - & 0 & 0 & - & 0 & 0 & 0 \\ 0 & - & 0 & X & X & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{array}{l} wvrD \\ lexA \\ umuD \\ recA \\ wvrA \\ wvrY \\ rvvA \\ polB \end{array}$$

Fig. 9. W identified after 50 learnings with $\lambda = 100$ on the first data set. Discretization process is described in the article. The j^{th} column shows all identified regulations exercised by j^{th} gene on other genes. Inversely, the i^{th} row shows all regulations the i^{th} gene is submitted to. Genes order is on the right.

Figure 9 shows the identified discretized W matrix using the first data set. One can notice that 9 probable regulations are identified in the network, which leads to an average degree of 2.25, as said previously. Inhibitions of *lexA*, *recA* and *wvrA* by *lexA* itself are well identified (second column). The activation of *lexA* by *recA* is also identified (W_{24}). False regulations due to *umuD* and *wvrA* (columns 3 and 5) are identified (these genes are target genes, and do

not act as regulators). A unknown regulation of *wvrA* by *recA* is identified: it could correspond to the indirect regulation $recA \rightarrow RecA \dashv LexA \dashv wvrA$.

For further comparisons between an identified network and the actual network, we have to specify the perfect W matrix which should correspond to the *S.O.S.* network. Of course, as some regulations are not directly transcriptional (for example the activation of *wvrA* by *recA* seen previously) the choice is not immediate. We shall consider that all parameters of the second column, indicating inhibitions on all genes by *lexA* have to be discretized in the class [-], while the activation of *lexA* by *recA* is represented by W_{24} in [+]. Other parameters of the fourth columns are not defined precisely and can be either in [0] or [+], because the learning is likely to identify undirect regulations.

Structure extraction can be viewed as an information retrieval task. We can transpose *Recall* and *Precision* measures usually used as quality measures in document retrieval to our field of interest. *Recall* defines the number of true oriented interactions predicted as fraction of all existing interactions. *Precision* defines the number of true oriented interactions predicted as fraction all the interactions predicted. Precision thus defines the level of "noise" in the information presented to the user.

But, it should be emphasized that structure extraction not only aims at finding regulations, but also at identifying absence of interactions. We should hence also measure the number of true predicted coefficients as fraction of all interactions between genes (regulations and non regulations). We define this number as the *Generalized precision*.

Recall mean on all 4 data sets is 0.66, with a standard deviation of 0.056; *Precision* gives a mean of 0.57, with a standard deviation of 0.083; *Generalized precision* is 0.87 associated with a standard deviation of 0.023. For a random matrix, one should notice that $Recall = 0.60$, $Precision = 0.20$, and $Generalized\ precision = 0.31$. *Generalization* and *Generalized performance* are significantly high. *Recall* seems to be quite low, but we have to remember that the true "score" of our method is given by the *Generalized precision*, as biologists are not only interested in regulations, but also in the absence of regulations. However the value of the *Recall* needs a comment: it can be enlightened by the fact that the experimental UV light shock was not sufficient to lead to the functioning of all *S.O.S.* genes. Figure 6 (top) shows that several genes were not induced during the experiment. These genes are activated only when the damage is sufficiently high.

In order to evaluate the similarity between networks identified using the various data sets, the proportion of equal coefficients between these networks are computed, giving a similarity mean of 89 % with a standard deviation of 11 %. It should be emphasized that these similarities are high, comparing with 25 % obtained with a random matrice. These high similarities show that several experiments on the same underlying network let similar networks be identified.

5.6 Adding a missing variable

The *S.O.S.* DNA repair of *E. coli* considered here involve both genes and some of their products, proteins. The fact that neither the level of *LexA* nor the level of *RecA* protein are available makes impossible to retrieve the true direct regulation pathways. It is especially worrying in the case of *LexA* which plays a central role. That is the reason why we decided to use one important feature of both the inertial model and the EM algorithm, the addition of a missing variable. When one missing variable is introduced, simulations of its level evolution are done using each of the 50 learned models. For 22 of these models, it is quite noticeable that the simulated profile is akin to the *LexA* protein concentration profile under the same experimental conditions measured in Sassanfar90 (see figure 10

Fig. 10. Simulated profile of the 9 nodes in the learned network and profile of *LexA* protein measured in similar conditions

Moreover, when considering only this cluster of models, variances concerning the regulations involving the added missing variable are lower than previously, so that W discretization shows that missing variable is inhibited by *recA* and inhibits *lexA* and itself. The obtained network is presented in the figure 11. g:SOS1missing

An attractive hypothesis is that the added missing variable "takes the role" of the protein *LexA*. Further experiments need to be done to show if this hypothesis is acceptable.

5.7 Prediction

Within the machine learning theory, the choice of a model and the identification of its parameters should lead to generalization ability. For sequential data, this ability can be measured by two properties: the model ability to make k -step ahead prediction and its ability to reflect dynamics of other i.i.d. sequences. In the context of the available data, our learned model easily succeeded in making k -step prediction using the first 2/3 data points for training and the 1/3 lasting time for prediction. This does not prove very much since prediction is quite easy (back to equilibrium) but is needed to be checked.

We also used one time course as training data and others as test data. Data provided by Alon are particularly adapted to this type of experiments, because the four available time-series concern the same network. Two kinds of prediction abilities are evaluated. The first one is called the *one time step prediction*: for each instant, an expectation of the observation at the next instant is computed using the test data observation and the model learned from the training data. The second prediction ability is called the *multi step*

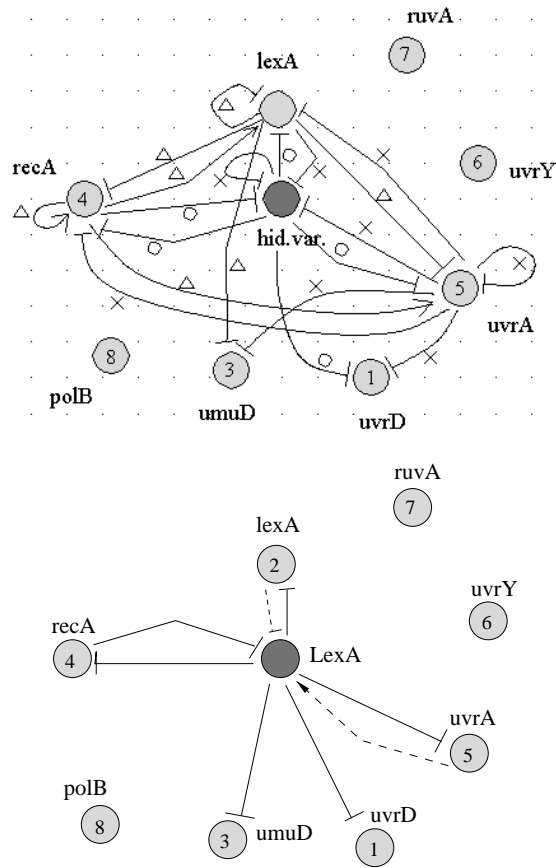


Fig. 11. Graph of gene interactions obtained after learning (left) and after learning+pruning (right)

prediction: a filtering phase is done on the 10 first instants of the testing data to estimate its hidden state at instant 10, and the model learned from the training data is then used to predict the kinetic profiles of the genes until instant 50 without accessing to their true observation values.

Very good correlations between *one step predicted* sequences and actual sequences are achieved, with a mean of 0.968, and a standard deviation of $15.6 \cdot 10^{-3}$. Correlation between the last part (instants 11 to 50) of *multi step predicted* sequences and actual sequences has a mean of 0.654 and a standard deviation of 0.171. Even if the predicted part of the sequence is the easiest because of the monotonous decrease of gene profiles after their peak, such correlations prove that the learned models are able to predict further evolution.

6 Conclusion and Future Work

We presented a general methodology for reverse modeling based on dynamical modelling and machine learning tools. We instantiate this approach to a new model of dynamics that yet share the advantages and the defaults of additive models while keeping into account delays and inertia in the response of a regulated gene. The first results on the *S.O.S. DNA Repair* network of *E. coli* were promising by showing the power of our approach and of the considered model. However this work could be improved in several ways. First, from the modeling point of view, the biological interpretation of the mathematical model we proposed can be further discussed. The assumption of coupled oscillatory behaviours even dampened for genes is at this stage an artifice of mathematicians. It allows us to capture complex behaviours whiel keeping a linear network. A future work will consist on studying other kinds of regulations where more belief can be put in the oscillatory assumption. Regulation of circadian rythms for instance can be an example of such regulations. Moreover, the inertial model belongs to the additive models family and does not take into account conjunction of regulations on the contrary of some models kile Savageau's one [?].

As a second direction, we want to bring elements of answer to a crucial point for the biologists : how many measurements should be produced to make possible the identification process ? In our time-serie framework, it reduces to the length of the avalaible time-courses. To our knowledge this question has not yet been solved even for linear dynamic models. It has to be taclked within the theory of statistical learning[?] and its recent results that link sample complexity to the model complexity. The principle is to get bounds on the size of the data sets that allow good prediction ability.

A third direction is to attempt to scale the learning method to large number of genes. At this stage for large networks, our learning algorithm meets too many local minima. Applying a “divide and conquer principle” is a possible way to solve this question. Some authors have already used this approach to reduce large networks to a set of connected subnetworks for each of which learning is easier (see [?],[?] for instance). Another way to address the question consists in taking into account more biological knowledge. Gene annotation can provide at this stage many informations such as gene functions and promotor sequences. A full bayesian approach applied to the dynamical bayesian network could be of help for incorporating such prior knowledge.