Bioinformatics, 34(7), 2018, 1253–1260
doi: 10.1093/bioinformatics/btx730
Advance Access Publication Date: 22 November 2017
Review

OXFORD

Systems biology

Extreme learning machines for reverse engineering of gene regulatory networks from expression time series

M. Rubiolo[1,2,*], D. H. Milone[1] and G. Stegmayer[1]

[1]Research Institute for Signals, Systems and Computational Intelligence, sinc(i), FICH/UNL-CONICET, Ciudad Universitaria, 3000 Santa Fe, Argentina and [2]Center of Research and Development of Information System Engineering, CIDISI, System Engineering Department, UTN-FRSF, 3000 Santa Fe, Argentina

*To whom correspondence should be addressed.

Associate Editor: Cenk Sahinalp

Received on March 3, 2017; revised on October 26, 2017; editorial decision on November 6, 2017; accepted on November 21, 2017

Abstract

Motivation: The reconstruction of gene regulatory networks (GRNs) from genes profiles has a growing interest in bioinformatics for understanding the complex regulatory mechanisms in cellular systems. GRNs explicitly represent the cause–effect of regulation among a group of genes and its reconstruction is today a challenging computational problem. Several methods were proposed, but most of them require different input sources to provide an acceptable prediction. Thus, it is a great challenge to reconstruct a GRN only from temporal gene expression data.

Results: Extreme Learning Machine (ELM) is a new supervised neural model that has gained interest in the last years because of its higher learning rate and better performance than existing supervised models in terms of predictive power. This work proposes a novel approach for GRNs reconstruction in which ELMs are used for modeling the relationships between gene expression time series. Artificial datasets generated with the well-known benchmark tool used in DREAM competitions were used. Real datasets were used for validation of this novel proposal with well-known GRNs underlying the time series. The impact of increasing the size of GRNs was analyzed in detail for the compared methods. The results obtained confirm the superiority of the ELM approach against very recent state-of-the-art methods in the same experimental conditions.

Availability and implementation: The web demo can be found at http://sinc.unl.edu.ar/web-demo/elm-grnnminer/. The source code is available at https://sourceforge.net/projects/sourcesinc/files/elm-grnnminer.

Contact: mrubiolo@santafe-conicet.gov.ar

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Reverse engineering of gene regulatory networks (GRNs) has been an intensively studied topic in bioinformatics since gene expression data began to be analyzed. Revealing the complex structure of regulation of gene expression by computational methods is a challenging task that has emerged in the last decades. Identifying subjacent genes interactions yields to an understanding of the topology of GRNs and the role of each gene, which is crucial to study the complex mechanisms underlying the behaviors of cells living in an environment (Hache *et al.*, 2009).

In recent years, several computational methods were proposed for the discovery of GRN from genomics data (Chai *et al.*, 2014; Ponzoni *et al.*, 2014). Within correlation methods (Faith *et al.*, 2007; Margolin *et al.*, 2006; Zhang *et al.*, 2012), TD-Aracne (Zoppoli *et al.*, 2010) adopts mutual information to compare expression profiles of gene expression data and applies different

network pruning heuristics. It assumes that a single perturbation experiment is performed, for example a treatment with a compound, gene knock down/up, etc. However, in most cases TD-Aracne was not able to detect the direction of a regulation. Among Bayesian models (Friedman *et al.*, 2000), BANJO (Yu *et al.*, 2004) infers gene networks from steady state and/or multiple time series of gene expression data. A system of linear ordinary differential equations are used in TSNIF (Polynikis *et al.*, 2009) for describing the regulatory network by modeling the interaction process among its various components, requiring a very large and detailed knowledge of the biological system under study and a huge computational power. Regarding formal methods, FormalM (Ceccarelli *et al.*, 2014) is a reverse engineering approach, which starts with a formal specification of gene regulatory hypotheses, mathematically proves whether a time course experiment belongs to the formal specification and finally determines whether a specific gene regulation exists or not. Neural networks (NNs) have been proposed as well (Noman *et al.*, 2013). A recurrent NN is trained (Mandal *et al.*, 2016) by using a hybrid Cuckoo Search (CS)-Flower Pollination Algorithm (FPA), to select the best combination of genes, but it sacrifices computational time complexity due to the hybrid optimization process used.

Most of the mentioned methods require several input sources to provide an acceptable network inference or discovery. Indeed, not only gene expression but also more experimental results such as the wild-type unperturbed network, the steady-state levels of single-gene knockouts and knockdowns are required as well, to provide a result. Thus, our work addresses an important challenge, which is being able to reconstruct a GRN by using only temporal gene expression data, modeling the regulations in a more accurate way and preserving also a low computational cost. A related work using time measures and Multilayer Perceptron (MLP) networks to model gene interactions has been recently proposed (Rubiolo *et al.*, 2015). However, it suffered from false positives and missing relations, due to the low generalization capacity of MLP models. Moreover, the use of this type of neural model generally results in a high computational cost since the classical backpropagation training algorithm (based on the optimization of the squared error) is very slow until convergence is achieved. Instead, our proposal here faces the mentioned challenge with a special type of more recent and better artificial NN named Extreme Learning Machine (ELM) (Huang *et al.*, 2006a). In this new proposal, the hyperparameters of the model can be automatically tuned beforehand and do not need to be exhaustively searched for during training. ELMs have also the capability of being extremely fast to be trained with high generalization capacity. Another difference with the previous proposal lies in the fact that, after the training of a large number of very slow MLP neural models in the study of Rubiolo *et al.* (2015), there were many rules for refining the relations to choose and many possible orders of applications among them. Depending on their combinations, different results can be achieved. Instead, in this new ELM-GRNNminer, the mining rules are much more simpler, straightforward and transparently applied. Moreover, the essence of ELM is that the learning parameters of hidden nodes are randomly assigned and need not to be tuned, while the output weights can be analytically determined by a simple generalized inverse operation. It has been proved that by utilizing ELMs, learning becomes very fast and with good prediction performance in many different fields (Huang *et al.*, 2015).

We present here an approach capable of effectively modeling all possible gene-to-gene regulations to reconstruct a GRN by using ELMs. The results obtained with artificial and real data show that the subjacent GRN can be effectively obtained only considering one input: the gene expression time series. Data from wild-type unperturbed network, the steady-state levels of single-gene knockouts and knockdowns, among others, are not necessary in the proposed method. Comparative results have shown better performance than state-of-the-art competitors.

2 Materials and methods

In this section, a description of a GRN mining approach for modeling the pairwise interaction between genes is presented. Then, the ELM model is detailed. Finally, it is explained how ELMs can be used for modeling the relationship between time series.

2.1 Mining GRN from data

Temporal gene expression profiles represent genes activity observed over a number of time steps. They can be used to train NNs aiming to model each gene-to-gene relation in a GRN. To discover the GRN underlying the data, this approach proposes: (i) training a pool of neural models with all available genes profiles, (ii) determining each potential gene-to-gene relation in the GRN to be reconstructed and (iii) identifying the strongest relations.

The training of each NN aims to measure how reproducible is the relationship between two time series in particular. That is to say, how much modelable is each gene-to-gene relation. It implies building a neural model for each possible combination of regulator–regulated genes and training it with the corresponding time series. This is based on the premise that if the training error is high after a large number of training epochs, it can be inferred that there is no possible relation between those genes; while if the error is low, it can be interpreted as an indication that such relation exists. When using the ELMs as predictors this way, the training error is directly used as indicator of whether a regulation exists. The error between the ELM prediction and the time series of the target gene is an indicator of how modelable is the particular relation under study. Each ELM is trained with all available data for the gene-to-gene regulation to be evaluated and the error is measured. Thus, ELMs are a mean to discover possible regulations.

To determine which gene is a potential regulator of another one, it is necessary to give a score to each relation found. This score can be based on how many times the smallest error has been achieved when modeling it, considering all the repetitions of each experiment, with random initializations (the number of repetitions is determined by calibration experiments with artificial data). This way, the ELM of the regulator that has the lowest error in most of the repetitions indicates which regulation is the easiest to model, in a statistical sense. This means that, for each regulated gene, it is possible to obtain a set of potential regulators with different probability of occurrence. Thus, for example, for gene A, an ELM model where gene B is possible regulator of gene A is trained many times (according to the number of repetitions). Also, another ELM model for gene C as possible regulator of gene A is trained the same number of repetitions. And so on, many ELM models are trained for all the possible regulators of gene A. For each repetition, the trained models are ordered by the lowest prediction error. A *score matrix* is built by accumulating in each regulated gene, for each repetition, 2 points for the regulator with the model with the lowest error and 1 point for the second one. After that, an *error matrix* is also built by obtaining the median error for all repetitions (random initializations) of the training procedure.

Then the identification of the real gene-to-gene regulation requires solving some dubious relationships between time series (i.e. possible regulator/regulated gene relationships), which could be false

positives. To resolve them it is necessary to apply some mining rules. Instead of the combination of many possible rules like in the study of Rubiolo *et al.* (2015), in this work only two refining rules are applied. The first one is the threshold rule, where weak connections are removed by setting a threshold below which only the strongest relations (lowest error) remain (the threshold is determined by calibration experiments with artificial data). The second one is the symmetry rule, where the true direction of regulation between two genes that seem to be mutually regulated is determined according to the relation between training errors and scores. Given two genes i and j, $\epsilon = (e_{ji} - e_{ij})/\max\{e_{ij}, e_{ji}\}$ relates the errors, for the ELM model in which i regulates j (e_{ij}) and the model where j regulates i (e_{ji}). When $\epsilon > 0$, $i \rightarrow j$ is the model with the lowest error. On the contrary, when $\epsilon < 0$ $j \rightarrow i$ is the best model. Analogously, $\rho = (s_{ij} - s_{ji})/\max\{s_{ij}, s_{ji}\}$ relates both model *scores*, s_{ij} and s_{ji}. Then, the average between ϵ and ρ is evaluated, and it can be inferred that $i \rightarrow j$ is the winning relationship when it is greater than zero (Rubiolo *et al.*, 2015). Finally, at the end of the application of these rules, the inferred GRN is obtained.

2.2 ELM-GRNNminer

ELMs have been proposed in the last decade as a novel learning algorithm for single hidden layer feedforward networks (SLFNs) (Huang *et al.*, 2006a). This method randomly chooses hidden nodes and analytically determines the output weights of SLFNs (Huang *et al.*, 2006b). The only free parameters that need to be learned are the connections (or weights) between the hidden layer and the output layer of the neural model. In this way, ELM is formulated as a linear-in-the-parameter model, which reduces to solving a linear system (Huang *et al.*, 2015).

For L arbitrary samples $(\mathbf{x}_\ell, y_\ell)$, in which $\mathbf{x}_\ell = [x_{\ell 1}, x_{\ell 2}, \ldots, x_{\ell N}]^T$ is considered the ℓth input vector and y_ℓ is the ℓth target output, the SLFNs output can be represented as

$$\widehat{y}_\ell = \sum_{i=1}^{P} \beta_i f(\mathbf{w}_i^T \mathbf{x}_\ell + b_i),$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{iM}]^T$ is the weights vector connecting the inputs to the ith hidden node, b_i is the bias of the hidden nodes, β_i is the weights connecting the ith hidden node and the output node, $f(\cdot)$ denotes activation function and P is the number of hidden nodes. The input weights $\mathbf{w}_i$ and the bias b_i are randomly assigned and never updated since they are a random projection. For reproducibility purposes, the same seed for the random generator can be used.

The training of the network aims to approximate the targets for the L input samples with zero error. That is, $\sum_\ell |\widehat{y}_\ell - y_\ell| = 0$, for which there must exist the appropriate $\widehat{\beta}_i$, $\widehat{\mathbf{w}}_i$ and $\widehat{b}_i$. It can be written compactly as $H\boldsymbol{\beta} = \mathbf{y}$, where the matrix $H = \{f(\widehat{\mathbf{w}}_i^T \mathbf{x}_\ell + \widehat{b}_i)\}$ has the outputs of the hidden layer for all the L input vectors, $\boldsymbol{\beta} = \left[\widehat{\beta}_1, \ldots, \widehat{\beta}_P\right]^T$ is the weights of the output layer and $\mathbf{y} = [y_1, \ldots, y_L]^T$ is the target outputs.

In the study of Huang *et al.* (2006a), it has been shown that the network parameters do not need to be fully adjusted when the activation function is infinitely differentiable. The input connection weights W and the bias can be randomly selected at the beginning of training, and the output connection weights can be obtained by solving $\arg\min_{\boldsymbol{\beta}} ||H\boldsymbol{\beta} - \mathbf{y}||$. Then $\widehat{\boldsymbol{\beta}} = H^\dagger \mathbf{y}$, where $H^\dagger$ is the Moore–Penrose generalized matrix inverse of H. Therefore, the learning algorithm of ELMs can be resumed in the following steps: (i) randomly assign the input weights W and the bias $\mathbf{b}$, (ii) calculate the outputs H of the hidden layer and (iii) obtain the weights $\widehat{\boldsymbol{\beta}}$ of the output node.

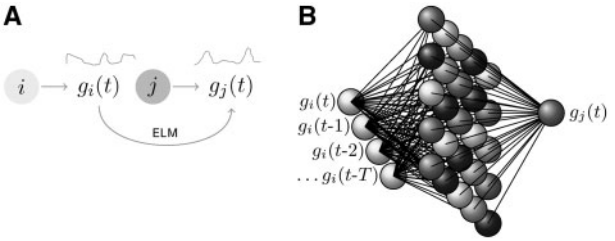

Fig. 1. Modeling a gene-to-gene regulation with ELMs. (**A**) A possible g_i-to-g_j regulation that can be modeled with an ELM. (**B**) ELM random projections from g_i-to-g_j time series data, by considering T time delays of g_i at the input

Compared with the traditional NN, the ELM does not need to adjust of W and $\mathbf{b}$ in the training process. It can obtain a global optimal solution only by fitting the $\boldsymbol{\beta}$, and this way the training speed can be improved significantly, avoiding to fall into local optimum (Huang *et al.*, 2006a,b).

In this work, a possible regulator–regulated relation is modeled by presenting all available gene expression time series to an ELM (Fig. 1). The original regulated time series and a time-delayed version of the regulator profile are shown to the ELMs (Fig. 1B), so the model can learn the temporal dynamics between these profiles. Considering the prediction error of the ELM as a measure of how much modelable is the gene-to-gene relation under analysis, it is possible to detect if there is an effective regulation between each pair of genes. The prediction capability of each ELM is measured with its mean square error. If this error is low (e.g. $<10E^{-6}$), it is a clear indication that such time relation (between a regulator and a regulated gene) exists.

Differently from the work of Rubiolo *et al.* (2015), the number of hidden neurons of the ELM can be automatically determined by generating a set of different topology networks and selecting the best with artificial data (detailed explanation in Section 2.3). This constitutes an important improvement over GRNNminer (Rubiolo *et al.*, 2015), in which a pool of MLP models were trained with different values for each of the possible parameters, resulting in a high number of experiments and, consequently, in high computational cost. An independent set can be previously used to obtain the hyperparameters of the ELMs. The time series from artificial GRNs can be used to estimate the optimal number of hidden nodes for a given network size, and then real GRNs can be used to measure the performance in terms of the gene regulations correctly discovered. Therefore, each possible regulation is modeled by a single ELM, which certainly contributes to the reduction of computational effort.

In ELM-GRNNminer there is a count on how many times, in relation to the repetitions with random initializations, the possible regulation between two genes, for example A → B has been found winner in relation to C → B. Furthermore, it is possible to find multiple regulations of a target gene, regardless that each ELM models only one gene-to-gene regulation, if both ELM models pass the error threshold. This way, both regulations (A → B and C → B) would be included in the reconstructed GRN. Each ELM could have received instead the time series for two input genes and one for one target gene; also three input genes could have been used for one target and so on, N input genes for one target. However, this way there would be scalability issues because all possible combinations should be considered as inputs and a very large number of ELMs should be trained. With ELM-GRNNminer, each ELM can be trained

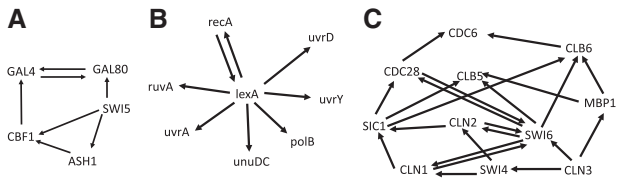

Fig. 2. Real GRNs. (**A**) IRMA (Cantone *et al.*, 2009), (**B**) *E.coli* SOS pathway (Ronen *et al.*, 2002) and (**C**) *S.cerevisiae* cell cycle (G_1 step) (Zoppoli *et al.*, 2010)

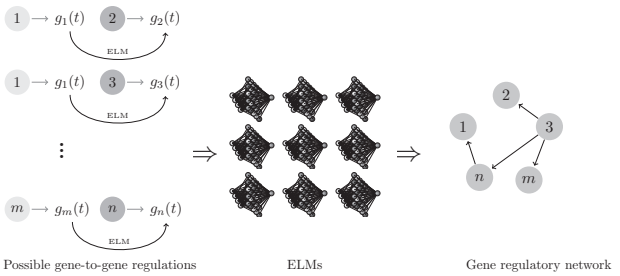

Fig. 3. Obtaining the GRN from several ELMs, which try to model each gene-to-gene regulation existing in the reconstructed GRN at the right

independently on multiple threads/servers in a very simple way (one ELM per processor). Thus, the model proposed here is actually very easily parallelizable and the scalability issue can be directly avoided on larger datasets.

2.3 Materials and experimental setup

Datasets used in the experiments are presented in this section. First, the artificial dataset used for automatically tuning the ELM parameters is described. Second, the three public experimental datasets used for analyzing the performance of the approach on real cases are presented. Finally, the experimental setup is defined.

2.3.1 Artificial datasets

The artificial data were produced by running GeneNetWeaver (http://gnw.sourceforge.net/genenetweaver.html), a recent open-source tool widely used for *in silico* benchmark generation and performance evaluation of regulatory network inference methods (Schaffter *et al.*, 2011). GeneNetWeaver adopts a kinetic network model of ordinary and stochastic differential equations and it is used in the DREAM network inference challenge, a competition where the goal is to predict network connectivities from gene expression datasets.

We generated 10 random networks of n genes topology, where $n = 5, 8, 11$, that is the number of genes involved in the experimental datasets. For each random network we generated experiments of T time points each, where T corresponds to the number of available time points in the experimental datasets. The 5 genes network had series of 21 time points; the time series of 8 genes network had $T = 14$ points and the 11 genes network had $T = 18$ points in their time series. Simulated time series were generated following the DREAM4 (http://dreamchallenges.org/project/dream4-in-silico-network-challenge/) challenge procedure, consisting of wild-type experiments without perturbations (knockouts, knockdowns, multifactorial or dual knockouts). For simulations, DREAM4 uses additive noise from a widely accepted model of noise, similar to a mix of normal and lognormal noises.

2.3.2 Real datasets

We have tested our novel approach in three real datasets. In vivo Reverse-engineering and Modeling Assessment (IRMA) is an *in vivo* fully controlled experimental network built in *Saccharomyces cerevisiae* and it is composed of five genes (Cantone *et al.*, 2009), connected as shown in Figure 2A. The SWI5 mutant is built by cloning upstream the promotor GAL1-10, which is able to activate transcription of the five genes network in the presence of galactose. Transcription of network genes was tested perturbing culturing cells by their shifting from glucose to galactose media and from galactose to glucose again. Dataset contains 16 and 21 time points, respectively, for each gene.

Escherichia coli SOS pathway is an eight genes network regulating the SOS response of DNA damage connected as in Figure 2B. This pathway is activated after *E.coli* cells are exposed to DNA damage agents and involves the induction of about 30 genes, many of which are related to DNA damage tolerance and repair. In our experiments, we consider the first 14 points time course profiles of the genes involved in DNA damage tolerance and repair activated through recA and lexA (Ronen *et al.*, 2002).

An 11 genes network from G_1 step of the cell cycle in *S.cerevisiae* is shown in Figure 2C. Only those genes whose mRNA levels respond to the induction of the two well-characterized cell cycle regulators Cln3 and Clb28 were chosen (Zoppoli *et al.*, 2010). Time course profiles (Spellman *et al.*, 1998) has 18 points.

2.3.3 Experimental setup and performance measures

To obtain which are the strongest regulations within all possibilities, each possible pair of times series data is used to train an ELM (Fig. 3). By analyzing the training error of each model, as explained in Section 2.1, it is possible to reconstruct the GRN. The number of hidden neurons at each ELM model can be determined by analyzing the results over artificial datasets automatically, in a transparent way to the user, differently from Rubiolo *et al.* (2015) where a range of values had to be determined and explored. In ELM-GRNNminer, an artificial dataset for calibration experiments was obtained, as explained in Section 2.3.1, with GeneNetWeaver from the DREAM challenge, which is nowadays the benchmark for GRN inference methods. Considering the (known) artificial GRN as a gold standard, the best possible ELM-GRNNminer hyperparameters were determined. For each GRN with different number of genes, the optimal threshold, time delay and number of hidden units were obtained in these calibration experiments. Regarding the repetitions number, this value was 10 for all of the cases, because the results in the calibration process did not change when the number of repetitions increased beyond this value. With the real datasets ELM-GRNNminer is used to obtain the predicted GRN and the true positives, true negatives, false positives and performance measures are calculated. These measures are not calculated on the times series used to train the ELM models. Thus, cross-validation with time series is not necessary because the ELM weights are not part of the GRN reconstructed.

To estimate the performance of the method we used the recall (R), specificity (S), precision (P), accuracy (A) and F-measure (F), defined as follows:

$$R = \frac{T_P}{T_P + F_N}, \quad S = \frac{T_N}{T_N + F_P}, \quad P = \frac{T_P}{T_P + F_P},$$
$$A = \frac{T_P + T_N}{T_P + T_N + F_P + F_N}, \quad F = 2\frac{PR}{P + R}$$

where T_P is the number of predicted edges that are correct (True Positives), F_P is the number of predicted edges that are wrong (False

Positives), T_N is the number of inexistent edges that are not found indeed and F_N is the number of true edges that were not predicted (False Negatives). Recall (or sensitivity) measures the ability of each method to identify overall positive results. Specificity measures the proportion of negatives that are correctly identified. Precision indicates the capability of the method for detecting the positives results with respect to true and false positives. Accuracy allows us to evaluate true cases (both positives and negatives) in the population. Finally, the *F*-measure estimates overall predictive capability of the method. It allows to better evaluate the effectiveness of the method in minimizing false values, both positives and negatives. Note, however, that the *F* does not take true negatives into account.

3 Results and discussion

To perform a comparative study we have chosen a set of related state-of-the-art reverse engineering methods recently proposed in literature, which are based on different computational approaches: BANJO (Yu *et al.*, 2004), TD-Aracne (Zoppoli *et al.*, 2010), TSNIF (Polynikis *et al.*, 2009), FormalM (Ceccarelli *et al.*, 2014), GRNNminer (Rubiolo *et al.*, 2015) and CS-FPA (Mandal *et al.*, 2016). Two versions of the proposed ELM-GRNNminer, with and without the applications of the rules described in Section 2.1, have been included in the comparisons.

Performance according to accuracy, specificity, precision, recall and *F*-measure is reported for the mentioned methods over IRMA (Table 1), *E.coli* (Table 2) and *S.cerevisiae* (Table 3) datasets. Precision and recall for BANJO, TD-Aracne, TSNIF and FormalM are those reported in the work of Ceccarelli *et al.* (2014). From these values, their corresponding accuracy, specificity and *F*-measure were deducted. CS-FPA values were obtained from its original work (Mandal *et al.*, 2016), in which it was evaluated only over *E.coli* dataset. Performance measures for GRNNminer were obtained by running the original implementation of the algorithm.

3.1 State-of-the-art comparison on real GRNs

Table 1 shows the results of the comparison for all the methods evaluated on IRMA dataset. Methods are listed in the first column. Performance measures are shown from second to sixth column. In terms of accuracy (second column), ELM-GRNNminer obtains the best performance. Regarding specificity, it can be noticed that ELM-GRNNminer (0.833) is slightly below FormalM, GRNNminer and TSNIF, but markedly above of BANJO and TD-Aracne. Considering now the precision of all compared methods, our method is positioned below TSNIF, at the same position than FormalM, and above all the others. Regarding recall (fifth column), ELM-GRNNminer has obtained the best performance (0.857). In global terms of *F*-measure (last column), the proposed method has the best performance.

From these results in IRMA dataset, it can be stated that ELM-GRNNminer is clearly superior to BANJO and TD-Aracne in all cases. Regarding TSNIF, ELM-GRNNminer has better results in term of all measures, except for precision. However, our method provides less false negatives, which has a positive effect by increasing the recall and the global accuracy. Even though FormalM has the same precision than ELM-GRNNminer, and a better specificity as well, in terms of *F*-measure our proposal is superior because of its higher recall. Regarding the GRNNminer in particular, although it is slightly superior in terms of specificity, for the rest of the measures our novel proposal is better, especially in the *F*-measure. This shows the effective improvement obtained by modeling the relations between time series with ELM models. Moreover, if ELM-GRNNminer is compared against its own version without mining rules (sixth row in Table 1), the former is superior most of the times. This indicates that the application of the mining rules raises the overall predictive performance of the method, in spite of incrementing the number of false negatives (as indicated by *R* and *F*).

Table 2 presents the performance on *E.coli* dataset. The accuracy (second column) shows that, although ELM-GRNNminer (0.875) is slightly below FormalM (0.906), it is markedly superior than the others. Similarly, in terms of specificity, it can be seen that ELM-GRNNminer has much better results than most of the methods. In this case it is overcome by FormalM (1.000) and GRNNminer (0.893), but at a high price: these two methods have a very low recall. The same trend is confirmed regarding precision (fourth column), in which ELM-GRNNminer is again above all the rest, only below FormalM that again has maximum precision but at the cost of an extremely low recall. Considering recall, our proposal obtains the best value (1.000) when no mining rules are applied. However, if the rules are applied, ELM-GRNNminer (0.875) shares the best recall with TD-Aracne, being highly superior than all the other

Table 1. Results of performance on IRMA dataset

	A	*S*	*P*	*R*	*F*
BANJO (2004)	0.640	0.647	0.455	0.625	0.526
TSNIF (2009)	0.800	0.882	0.714	0.625	0.667
TD-Aracne (2010)	0.400	0.235	0.316	0.750	0.444
FormalM (2014)	0.800	0.889	0.667	0.571	0.615
GRNNminer (2015)	0.720	0.889	0.500	0.286	0.364
ELM-GRNNminer (without rules)	0.680	0.611	0.462	0.857	0.600
ELM-GRNNminer	0.840	0.833	0.667	0.857	**0.750**

Best scores in bold.

Table 2. Results of performance on *E.coli* dataset

	A	*S*	*P*	*R*	*F*
BANJO (2004)	0.734	0.750	0.263	0.625	0.370
TSNIF (2009)	0.688	0.750	0.125	0.250	0.167
TD-Aracne (2010)	0.203	0.107	0.123	0.875	0.215
FormalM (2014)	0.906	1.000	1.000	0.250	0.400
GRNNminer (2015)	0.797	0.893	0.143	0.125	0.133
CS-FPA (2016)	0.766	0.804	0.267	0.500	0.348
ELM-GRNNminer (without rules)	0.828	0.804	0.421	1.000	0.593
ELM-GRNNminer	0.875	0.875	0.500	0.875	**0.636**

Best scores in bold.

Table 3. Results of performance on *S.cerevisiae* dataset

	A	*S*	*P*	*R*	*F*
BANJO (2004)	0.669	0.765	0.207	0.261	0.231
TSNIF (2009)	0.777	0.936	0.500	0.222	0.308
TD-Aracne (2010)	0.529	0.510	0.226	0.609	0.329
FormalM (2014)	0.438	0.387	0.230	0.607	0.333
GRNNminer (2015)	0.727	0.878	0.143	0.087	0.108
ELM-GRNNminer (without rules)	0.545	0.561	0.204	0.478	0.286
ELM-GRNNminer	0.653	0.694	0.268	0.478	**0.344**

Best scores in bold.

Table 4. Overall comparison of the performance (*F*-measure) of state-of-the-art methods against ELM-GRNNminer over three experimental datasets: IRMA, *E.coli* and *S.cerevisiae*

	IRMA	*E. coli*	*S. cerevisiae*
BANJO (2004)	0.526	0.370	0.231
TSNIF (2009)	0.667	0.167	0.308
TD-Aracne (2010)	0.444	0.215	0.329
FormalM (2014)	0.615	0.400	0.333
GRNNminer (2015)	0.364	0.133	0.286
CS-FPA (2016)	—	0.348	—
ELM-GRNNminer	**0.750**	**0.636**	**0.344**

Best scores in bold.

methods. *F*-measure indicates that the proposed ELM-GRNNminer has the best global performance, being markedly superior to all other methods.

From Table 2, it can be stated that our proposal has better global prediction results than state-of-the-art method in this more complex GRN. Comparing TD-Aracne against ELM-GRNNminer, although both share the same recall, the rest of the performance measures are better in our case (noticeably in terms of *F*). Similarly, despite GRNNminer having a slightly better specificity than our proposal, it has a lower performance in the rest of the measures. Regarding the particular comparison of ELM-GRNNminer against FormalM, although the last one has better accuracy, specificity and precision, its recall is markedly inferior. This occurs because our proposal is capable of minimizing the number of false negatives in the resultant reconstructed GRN, which consequently improves the global performance, placing ELM-GRNNminer as the best method. It is important to highlight that ELM-GRNNminer without mining rules (seventh row in Table 2) obtains the best recall as well. This shows, again, that modeling the regulations with ELM and the posterior application of mining rules collaborates in terms of the global performance.

Table 3 shows the performance on the much more complex *S.cerevisiae* dataset. The global trend is similar than in the previous datasets. In terms of accuracy, specificity and precision, TSNIF obtains the best values. Looking at the recall alone (fifth column), TD-Aracne and FormalM are the best ones. However, again, those methods have a disappointing low global performance. ELM-GRNNminer has the best *F*-measure. If ELM-GRNNminer is compared with its version without the application of the mining rules (sixth row), it is possible to see that, although the last one achieves the same recall, the first one obtains higher performance in the other measures and, consequently, in the global performance. This supports, again, the need of the application of mining rules to refine the final solution.

To summarize, Table 4 shows the overall results for the comparison of the proposal against the state-of-the-art methods when they are applied over the three real datasets, according to the global measure *F*. In the case of IRMA, ELM-GRNNminer obtains the best result: 0.750. With regard to *E.coli* dataset, the proposed method has obtained the best result, 0.636, which is clearly superior that the next best (FormalM). At last, for *S.cerevisiae* the best *F* (0.344) is obtained again by ELM-GRNNminer. Considering these overall results, it can be stated that ELM-GRNNminer obtains the best result for all the three real datasets. It must be highlighted that none of the five state-of-the-art methods compared achieves the best performance in terms of the global *F*-measure, for all the three real

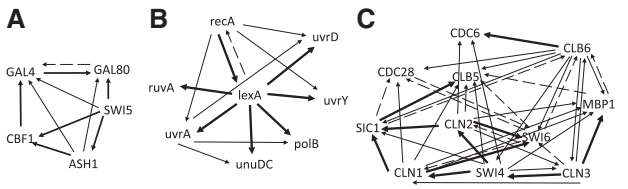

Fig. 4. Resultant GRN after the application of ELM-GRNNminer over IRMA (**A**), *E.coli* (**B**) and *S.cerevisiae* (**C**). Thick lines denote true-positive values, thin lines represent false-positive values and thin-dotted lines show the false-negatives results

benchmark datasets, as ELM-GRNNminer does. This global comparison shows the benefits of modeling the regulation between temporal gene profiles with ELMs, with an effectively reduced number of false negatives.

Finally, Figure 4 shows a detail of the performance of ELM-GRNNminer in terms of the resultant GRN in comparison with the gold standard, for the three real datasets. Thick lines show the true positives, thin lines show the false positives and thin-dotted lines the false negatives. In case of IRMA (Fig. 4A), it is possible to see that most true regulations are found, a few false positives are obtained and there is only one false negative. Regarding *E.coli* GRN (Fig. 4B), the same trend is identified: most of the true regulations, a few false regulations and only one false negative are found. At last, considering the *S.cerevisiae* GRN (Fig. 4C), although there are more false results than in the other datasets, the most important positives values are effectively discovered.

As can be seen from the graphical analysis, our proposal can be really useful to reconstruct a hidden GRN from data, in which the biologist can elaborate hypothesis, plan wet experiments and then confirm them experimentally. Moreover, it is easy to see that the number of false positives is actually very low, as has already been shown in the tables, which can have a great importance at the time of experimental validation.

3.2 Increasing GRNs sizes

To compare the state-of-the-art methods against ELM-GRNNminer for increasing GRN sizes, and to analyze the impact of this size on the performance of the methods, several artificial datasets from small to complex GRNs were generated with GeneNetWeaver (http://gnw.sourceforge.net/genenetweaver.html). Five GRN topologies were obtained with 10, 20, 30, 40 and 50 genes. For each GRN, 10 realizations of the time series were obtained by repeating the simulation. ELM-GRNNminer, BANJO, TD-Aracne and TSNIF were applied to these replicates for each GRN topology. The resulting GRNs were compared against the golden standard used for the simulation, and performance measures were computed.

Figure 5 shows the mean *F*-measure comparison for the state-of-the-art methods and our proposal, for all the GRN topologies. In the *x*-axis are the sizes of the GRNs, and the *y*-axis presents the mean *F*-measure. In all of the cases, it can be seen that ELM-GRNNminer has a superior performance over the other methods. For the 10-genes GRN, ELM-GRNNminer has obtained almost twice the performance in comparison with BANJO and even more in comparison with TD-Aracne and TSNIF. In the 20-genes GRN case, the gap is shorter between the mean *F*-measure obtained for ELM-GRNNminer, BANJO and TD-Aracne, but this difference is almost four times larger than TSNIF. For GRNs having 30 and 40 genes the trend is the same: ELM-GRNNminer is almost twice as better than BANJO and TD-Aracne, being these two close between them, and being TSNIF the worst method. The most complex 50-

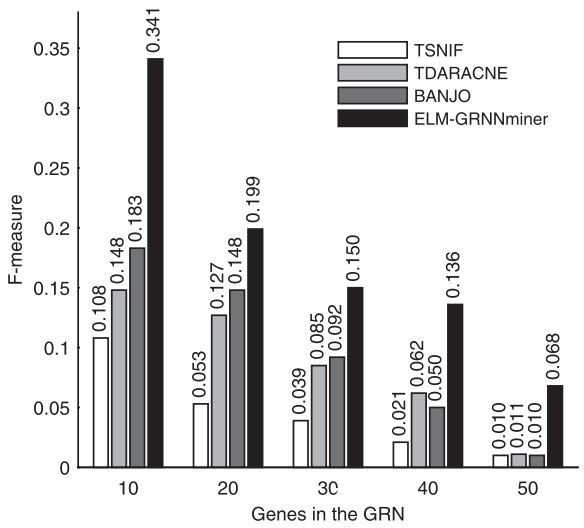

Fig. 5. *F*-measure comparison with state-of-the-art methods and ELM-GRNNminer for increasing GRNs sizes

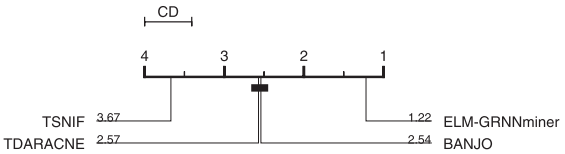

Fig. 6. CD diagram for *F*-measure in all datasets

genes GRN case shows that ELM-GRNNminer has again the best performance, and its distance to the other methods (among which there is almost no difference) is six times larger. Globally, all methods are affected when the complexity of the hidden GRN and number of genes involved in it increases. However, the performance drops about 5 times for ELM-GRNNminer, while it drops 10–20 times for the other methods. It can be seen that the comparative performance of the methods when increasing the network size is the same as for the previous section: ELM-GRNNminer has the best performance, followed by BANJO, TD-Aracne and TSNIF. The detailed performance measures for these experiments (precision, recall, accuracy and *F*-measure values, for each fold) can be seen in the Supplementary Material.

To statistically evaluate the performance differences between the methods tested, a Friedman rank test (Demsar, 2006) at significance level $\alpha = 0.05$ was carried out with *F*-measure in all datasets, giving a very low *P*-value ($1.0369e^{-19}$). This indicates that there are methods with statistically significant differences in the performance over the evaluated datasets. Then, the corresponding critical difference (CD) diagram for *post-hoc* Nemenyi tests on the results (Demsar, 2006) is shown in Figure 6, with a CD of 0.59. This clearly indicates that ELM-GRNNminer is the best method. The figure shows also that there are no statistically significant differences between BANJO and TD-Aracne. The worst method for this GRNs inference task was TSNIF, far distant from TD-Aracne/BANJO, and ELM-GRNNminer.

4 Conclusions

This work proposed a novel way to infer a GRN underlying gene profiles by modeling all possible gene-to-gene regulations by ELMs. The original regulated profile and a time-delayed version of the regulator profile are shown to the corresponding ELM, so the model can learn the temporal dynamics. Considering the prediction error of the ELM as a measure of how much modelable is a gene-to-gene relation, it was possible to detect if there is an effective regulation between each pair of analyzed genes. This way the high learning rate and best performance in terms of prediction capabilities of ELMs were exploited.

The proposal has demonstrated a better global accuracy than recent proposals, over real datasets. ELM-GRNNminer was the only method capable of obtaining the best results in all datasets for the challenging task of identifying the GRN just by considering gene expression data. Furthermore, the small false negatives rate can be a very useful advantage for the user because this guarantees a minimum loss of real regulations. Moreover, the tool had shown robustness and a superior performance over other recent methods when increasing the size of the GRNs to be discovered. Our approach has shown to be an effective and accurate tool to infer the hidden GRN only from time series of gene expression.

Funding

This work has been supported by the National Scientific and Technical Research Council (CONICET) [PIP 2013 117], National University of Litoral (UNL) [CAI+D 2011 548, 2016 076, 2016 082], National Technological University (UTN) [EIUTNFE0004442] and Agencia Nacional de Promocion Cientifica y Tecnologica (ANPCyT) [PICT 2014 2627].

References

Cantone,I. *et al.* (2009) A yeast synthetic network for *in vivo* assessment of reverse-engineering and modeling approaches. *Cell*, **137**, 172–181.

Ceccarelli,M. *et al.* (2014) *De novo* reconstruction of gene regulatory networks from time series data, an approach based on formal methods. *Methods*, **69**, 298–305.

Chai,L.E. *et al.* (2014) A review on the computational approaches for gene regulatory network construction. *Comput. Biol. Med.*, **48**, 55–65.

Demsar,J. (2006) Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, **7**, 1–30.

Faith,J.J. *et al.* (2007) Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles. *PLoS Biol.*, **5**, 1–13.

Friedman,N. *et al.* (2000) Using Bayesian networks to analyze expression data. *J. Comput. Biol.*, **7**, 601–620.

Hache,H. *et al.* (2009) Reverse engineering of gene regulatory networks: a comparative study. *EURASIP J Bioinform. Syst. Biol.*, **2009**, 617281.

Huang,G. *et al.* (2015) Trends in extreme learning machines: a review. *Neural Netw.*, **61**, 32–48.

Huang,G.-B. *et al.* (2006a) Extreme learning machine: theory and applications. *Neurocomputing*, **70**, 489–501.

Huang,G.-B. *et al.* (2006b) Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Netw.*, **17**, 879–892.

Mandal,S. *et al.* (2016) Large-scale recurrent neural network based modelling of gene regulatory network using cuckoo search-flower pollination algorithm. *Adv. Bioinformatics*, **2016**, 5283937.

Margolin,A.A. *et al.* (2006) ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, **7**, S7.

Noman,N. *et al.* (2013) *Reconstruction of Gene Regulatory Networks from Gene Expression Data Using Decoupled Recurrent Neural Network Model.* Springer, Tokyo, Japan, pp. 93–103.

Polynikis,A. *et al.* (2009) Comparing different ODE modelling approaches for gene regulatory networks. *J. Theor. Biol.*, **261**, 511–530.

Ponzoni,I. *et al.* (2014) Pathway network inference from gene expression data. *BMC Syst. Biol.*, **8**, S7.

Ronen,M. *et al*. (2002) Assigning numbers to the arrows: parameterizing a gene regulation network by using accurate expression kinetics. *Proc. Natl. Acad. Sci. USA*, **99**, 10555–10560.

Rubiolo,M. *et al*. (2015) Mining gene regulatory networks by neural modeling of expression time-series. *IEEE/ACM Trans. Comput. Biol. Bioinform*., **12**, 1365–1373.

Schaffter,T. *et al*. (2011) GeneNetWeaver: *in silico* benchmark generation and performance profiling of network inference methods. *Bioinformatics*, **27**, 2263.

Spellman,P.T. *et al*. (1998) Comprehensive identification of cell cycle regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell*, **9**, 3273–3297.

Yu,J. *et al*. (2004) Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, **20**, 3594–3603.

Zhang,X. *et al*. (2012) Inferring gene regulatory networks from gene expression data by path consistency algorithm based on conditional mutual information. *Bioinformatics*, **28**, 98.

Zoppoli,P. *et al*. (2010) TimeDelay-ARACNE: reverse engineering of gene networks from time-course data by an information theoretic approach. *BMC Bioinformatics*, **11**, 154.