Mining gene regulatory networks by neural modeling of expression time-series

Mariano Rubiolo, Diego Milone, *IEEE Member*, Georgina Stegmayer, *IEEE Member*

Abstract—Discovering gene regulatory networks from data is one of the most studied topics in recent years. Neural networks can be successfully used to infer an underlying gene network by modeling expression profiles as times series. This work proposes a novel method based on a pool of neural networks for obtaining a gene regulatory network from a gene expression dataset. They are used for modeling each possible interaction between pairs of genes in the dataset, and a set of mining rules is applied to accurately detect the subjacent relations among genes. The results obtained on artificial and real datasets confirm the method effectiveness for discovering regulatory networks from a proper modeling of the temporal dynamics of gene expression profiles.

Index Terms—Gene Regulatory Networks, Gene profiles, Times Series Data, Neural Networks.

1 Introduction

A GENE regulatory network (GRN) is an abstract mapping of gene regulations in living organisms that can help to predict the system behavior. During last years, many approaches have been proposed to unravel the complexity of gene regulation [1]. Genes interact with one another and these interactions can be measured over a number of time steps, producing temporal gene expression profiles. A hot topic on gene expression data analysis nowadays is the reconstruction of a GRN from such data, revealing the underlying network of gene-to-gene interactions [2]. In other words, the goal is to determine the pattern of activations and inhibitions amongst genes that make up the underlying GRN.

Given the expression levels of a set of interacting genes measured at different time points, formal methods can be developed to model gene interaction [3]. In fact, discovering gene regulatory networks by data-driven methodologies has been under study in the last years [4]–[7]. For instance, Boolean networks [8] [9] only consider the expression of a gene as on/off (do not taking into account intermediate expression level) hence having inadequate resolution. Bayesian networks [10] are represented as graphs considering the joint probability distribution of genes. This model can capture the inherent noise and stochastic behavior in gene expression data effectively, but the high computational cost hinders the application to a large number of genes in network. Dynamic Bayesian networks [11] have extended Bayesian networks to include temporal components. Also genetic algorithms [12] were applied to predict the regulatory pathways represented as an influence matrix. This approach can be applied with a small amount of data, optimizing a large number of parameters simultaneously, and solving nonlinear models. A neural network (NN) based model was presented in [13] for identifying gene regulatory networks from temporal gene expression data, in which both feedforward and Elman networks were used. The proposed models have a structural resemblance to genetic networks and also have the ability to capture the strengths of gene interactions in the form of network weights. Two iteratives methods were proposed to determine a set of potential regulators of a target gene. An artificial dataset that contained a subjacent GRN was used to test the approach, and some of the existing interactions were found, but not all of them.

Recently, the results of a comparative study [14] among six different reverse engineering methods, have highlighted the NN approach as the best performing method. This approach [15] involved a Recurrent Neural Network, whose topology was a simplified representation of a GRN with a weight matrix having non-zero entries to indicate interactions (activation or inhibition) of nodes (representing genes). Sensitivity was 57%, that is, only about half of the regulations were found.

Artificial neural networks can be used to infer genetic networks by modeling pairs of genes activity over a number of time steps. Thus, for modeling gene regulation all the possible combinations between genes must be analyzed in order to discover their relations. Using neural networks for modeling interactions in genetic networks requires training them to predict a target gene regulation from candidate regulating profiles. By adjusting their

- *M. Rubiolo is with CIDISI, UTN-FRSF, Lavaise 610, (3000) Santa Fe, Argentina, mrubiolo@santafe-conicet.gov.ar*
- *M. Rubiolo, D. Milone and G. Stegmayer are with sinc(i), UNL-CONICET, Ciudad Universitaria, Paraje El Pozo, RN 168 Km. 472.4, (3000) Santa Fe, Argentina.*

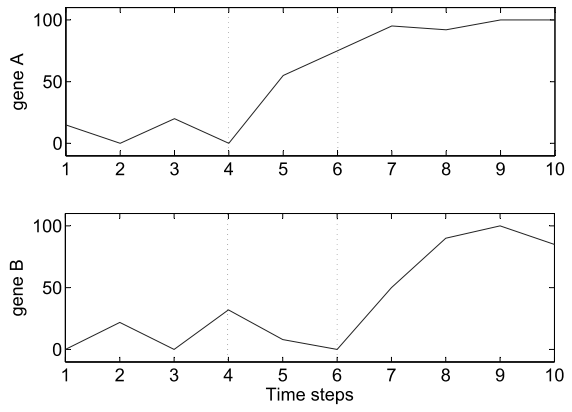

Fig. 1. Temporal delay identification between two gene expresion levels data.

synaptic weights, NNs alter their configuration to model each gene connection, which results in a minimum error in predicting a target profile.

This work proposes a novel approach to discover a GRN from temporal genes expression profiles by using a pool of multi layer perceptrons (MLP) with temporal delays at the input, named *GRNNminer*[1]. In any GRN, the complex interactions occurring amongst genes can be either instantaneous or time-delayed [16]. Genes expression profiles are considered here as time series [17], and are used to train a pool of NN models. Each NN is designed to discover, for a target gene profile at the output, the potential regulator of that gene at the input, by modeling their gene-to-gene interaction during a time period. The ability of each trained NN to model each possible relation is judged by the evaluation of the generalization error during the training process. For each possible relationship found, a score matrix is computed to detect the most likely regulations. Three rules for mining the unknown GRN from the score matrix are proposed, which allows to discover each correct gene-to-gene relation from all the possible ones.

The manuscript is organized as follows: Section 2 explains in detail the novel *GRNNminer* approach for GRN mining. Section 3 presents the datasets used in this study, whereas Section 4 explains the experimental setup and introduces the performance measures used for validation. In Section 5 the experimental results are presented and discussed. At last, conclusion and future works are given in Section 6.

2 Novel approach for GRN mining

This section explains how a NN can be trained by using temporal data from gene profiles for modeling gene-to-gene relations. After that, the novel *GRNNminer* will be presented for discovering the subjacent GRN.

1. The source code is freely available for academic purpuses at http://sourceforge.net/projects/sourcesinc/files/grnnminer/

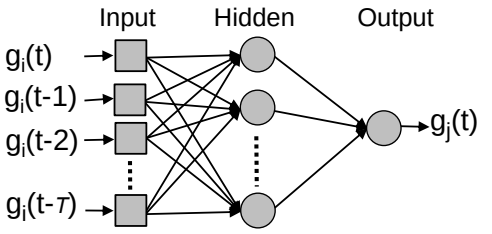

Fig. 2. NN model with temporal delays at the input.

Temporal gene expression profiles represent genes activity observed over a number of time steps. For instance, in Figure 1 a segment of two genes profiles is shown both possibly related. It can be seen that, for gene A at time 4 the signal starts to increase, while for gene B this behavior is only beginning to be perceived at the sixth time instant. It could be stated that gene A behavior has some influence on gene B, and the temporal delay between the activation of these two genes is around 2 time steps. This temporal series can be used to train a NN aiming to model this gene-to-gene relation in a GRN.

A NN can be loosely defined as a large set of simple interconnected units (neurons), which are executed in parallel to perform a common global task [18]. These units usually undergo a learning process that automatically updates network parameters (the connections among neurons, also named synaptic weights) in response to a possibly evolving input environment [19]. The model topology has several layers with no connections among neurons belonging to the same layer. The number of layers and neurons in each layer is chosen a priori, as well as the type of activation functions for the neurons [20].

The relation between two signals can be modeled by using a NN. According to the ability for modeling this relation within a bounded number of training epochs, it could be possible to detect a regulation between genes represented by two temporal series. Figure 2 shows a NN model with time delays at input layer, which can consider explicitly the temporal structure of the input signal. Specifically, the MLP has three layers (input, hidden and output), a sigmoid activation function at the hidden layer, and a linear activation function at the output. The first layer has only input connections. The basic learning algorithm used is backpropagation [20] which uses gradient descent to minimize a cost function, generally defined as the Mean Squared Error (MSE) between the desired output (targets) and the actual network output. During learning, the error propagates backwards through the network and the model parameters are changed accordingly [21]. The number of inputs is related to the number of temporal delays considered

TABLE 1
Example of error ranking after five repetitions of an experiment aiming to discover the regulator of gene B.

Repetition	*Regulator*	*Regulated*	*Error*
1	A	B	$5.21e^{-10}$
	C	B	$6.10e^{-10}$
2	A	B	$8.92e^{-10}$
	D	B	$9.57e^{-10}$
3	A	B	$0.35e^{-09}$
	E	B	$1.75e^{-09}$
4	A	B	$2.19e^{-09}$
	D	B	$8.32e^{-09}$
5	A	B	$0.98e^{-08}$
	C	B	$1.64e^{-08}$

(window τ). That is to say, there will be $\tau + 1$ inputs, considering the input signal from time t to $t - \tau$. The value of temporal delays τ considered at the input signal can be determined by analyzing each gene-to-gene relation. In this way, all the possible combinations of genes should be analyzed to determine the temporal delay for each dataset.

Discovering all the possible relations among a set of genes is the main objective of this network mining approach. In order to achieve this aim, each possible gene-to-gene relation can be modeled by a NN with temporal delays. This model has to be trained and the results must be analyzed to detect which is the most probable GRN underlying the dataset. The *GRNNminer* method consists in three steps: 1) modeling all possible gene-to-gene relations by using a pool of NNs, 2) putting a score to each relationship found, 3) applying rules to discover the underlying GRN. They are explained in detail in the following subsections.

2.1 Pool of NN for modeling gene-to-gene relations

Modeling gene-to-gene relationships implies building a NN for each possible combination of regulator regulated genes and training the model with the corresponding temporal series. In this work, it is important to find a good model that takes into account the dynamics underlying the data. In other words, how good is the NN to identify the dynamics of the model. If training error is high after a large number of training epochs, it is possible to conclude 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. However, if the relationship between two signals is not clear enough, independently of the number of training epochs, at the end of the training process the error will be high anyway. Similarly, if the relationship between two signals is clear enough, whatever the number of training epochs, the error will be low. In this work, the training error is calculated regarding the prediction of the regulated gene, as the mean square difference between the target time series (of the regulated gene) and the output time series obtained from the neural model.

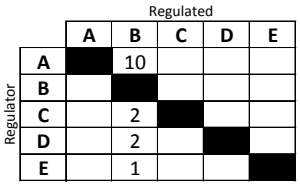

Fig. 3. Example of filling one column of the scoring matrix after considering Table 1.

The training and test of the models have to be repeated several times in order to achieve more robust statistical results. Therefore, there will be a set of MLP models specialized in each relation. Globally, this can be considered as a pool of NNs whose main objective is to measure how reproducible is the relationship between two temporal series. That is to say, how much modelable is each gene-to-gene relation. The ability of a trained NN to model a gene network is assessed using MSE, which is the mean of discrepancies between the gene expression predictions made by a NN and the true gene expression profiles, over all measurements. Therefore, for each training done within the pool of models, this generalization error is measured by cross-validation in order to find the differences and similarities between input and target genes. This error is considered in the next step to apply the scoring method.

2.2 Scoring method

In order to determine which gene is a potential regulator of another one, it is necessary to give a score to each modeled relation. This score is based on how many times the smallest error has been achieved, considering all the repetitions of each experiment, with random initializations. This way, the neural model that has the lowest error in most of the repetitions determines which regulation is the easiest to model, in a statistical sense. For instance, let us suppose we have five genes $\{A, B, C, D, E\}$. All the candidates as regulators of gene B are listed. Then, a score is given to each possible *regulator-to-regulated* relation $\{A \rightarrow B, C \rightarrow B, D \rightarrow B, E \rightarrow B\}$. This is carried out for all the genes, so that all alternatives are evaluated.

The results must be ordered so that the smallest error is at the top of the list whereas the largest at the bottom, as can be seen in Table 1. In order to clarify the explanation, in this Table only the first five repetitions (first column of the table) and the first two pairs with minimum MSE for each repetition are shown. In this example, the experiment which evaluates if gene A regulates gene B has the minimum MSE in all repetitions. After that, the scoring method is applied.

The scoring method consists in giving 2 points to the *regulator-regulated* relation with the lowest error, (the first position on the list), and 1 point to the following relation with the lowest error (the second position on the list). This procedure is iterated as many times as repetitions in the table. For instance, from the list shown in Table 1 it is possible to see that 5 repetitions have been done to determine which is the regulator of gene B. These repetitions are separated in the list by using a dotted line. In all of them, the relation $A \rightarrow B$ has the first position, so its score is 10. Relations $C \rightarrow B$ and $D \rightarrow B$ have the second position in two repetitions, therefore 2 points are assigned to both of them. At last, the score for $E \rightarrow B$ is 1 because it obtained the second position in only one repetition.

Once the points have been assigned, it is possible to build a scoring matrix to represent each possible gene-to-gene relations, as can be seen in Figure 3. Rows represent the potential regulator genes whereas columns depict regulated genes. The diagonal is not taken into account because autoregulation is not considered in this work. Filling the scoring matrix involves putting the scores on the corresponding *regulator-regulated* cell in the matrix. For instance, the second column of the matrix shown in Figure 3 is filled with the scores calculated above by applying the scoring method over the list.

2.3 Mining rules

During the previous step, the scoring matrix is filled with the values obtained after analyzing the results of the pool of NNs defined in Subsection 2.1. In Figure 4a), an example of a full scoring matrix is presented. It is possible that some doubtful relations can be present on the matrix. The aim of the rules is to solve these conflicts, as follows.

2.3.1 Threshold rule

Observing Figure 4a) it is difficult to identify which gene is regulated by whom because of the different score. It is necessary to eliminate weak connections from the score matrix in order to clarify the way the genes are related. Therefore, the first step consists in excluding those relationships whose errors are above a threshold. This threshold is calculated as the lowest error in all runs plus a percentage θ of that error. Considering the errors associated with the example, the grey cells in Figure 4a) are the relationships that must be removed. Figure 4b) shows the resultant matrix after the application of this rule.

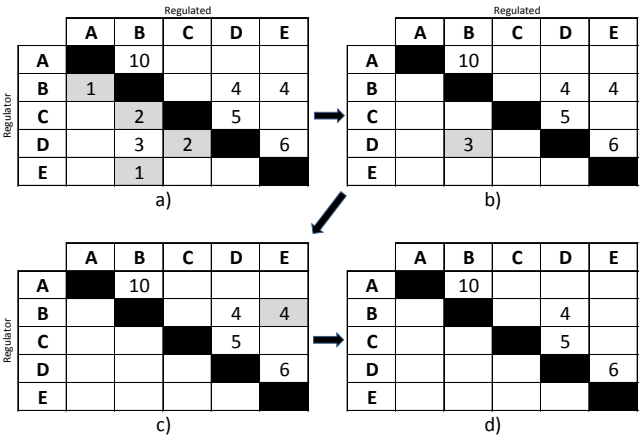

Fig. 4. Application of mining rules. a) *Original* scoring matrix. b) Resultant matrix after *Threshold* rule is applied. c) Resultant matrix after *Symmetric* rule is applied. d) Resultant matrix after the application of *Unchained* rule.

2.3.2 Symmetric rule

Another doubtful case can occur between two particular genes that are apparently mutually regulated. For instance, in Figure 4b) it can be seen how the regulation $B \rightarrow D$ has 4 points and the regulation $D \rightarrow B$ contains 3 points. After the application of this rule, shown in Figure 4c) , only remains the relation with the best average value between normalized error and score.

Given two genes i and j, ϵ relates both possible models errors as

$$\epsilon_s = \frac{e_{ji} - e_{ij}}{\max\{e_{ij}, e_{ji}\}}, \quad (1)$$

where e_{ij} is the error of the model $i \rightarrow j$, and e_{ji} is the error of the reverse model $j \rightarrow i$. When $\epsilon_s > 0$, it means that $i \rightarrow j$ is the model that has the lowest error. On the contrary, when $\epsilon_s < 0$ the $j \rightarrow i$ model has the lowest value.

Analogously, ρ_s relates both models *scores* as

$$\rho_s = \frac{s_{ij} - s_{ji}}{\max\{s_{ij}, s_{ji}\}}, \quad (2)$$

where s_{ij} is the score of the model $i \rightarrow j$, and s_{ji} is the score obtained by the reverse model $j \rightarrow i$, so that $\rho_s > 0$ if the first model has the maximum score. In both cases, ϵ_s and ρ_s are normalized in order to have the same weight in the average, by dividing each one by their corresponding maximum value.

Therefore, when the average between ϵ_s and ρ_s is greater than zero, it can be inferred that $i \rightarrow j$ is the winning relationship, and when the average is less than zero, the opposite $j \rightarrow i$ is true.

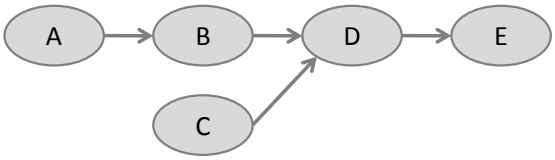

Fig. 5. A simple GRN discovered at the end of the mining rules application.

This approach could be used in more general cases where mutual regulation is present in the biological network simply by not applying this symmetric rule. In this way mutual regulations would be fully contemplated within the model.

2.3.3 Unchained rule

In Figure 4c) it can be observed another special case among genes *B*, *D* and *E*. Analyzing the matrix, the relations *B*→*D*, *D*→*E* and *B*→*E* may be present. It is necessary to remove one of the relations to unchain a possibly redundant regulation. Two possible scenarios can be considered here. The first one is that the correct chain is *B*→*D*→*E*, thus regulation *B*→*E* should be removed because *E* is regulated by *B* but through *D*. The second scenario could be that the correct regulations are *B*→*D* and *B*→*E*. In this case, *D*→*E* should be discarded because both genes *D* and *E* are regulated by the same parent gene *B*. To determine whether one situation or the other is the correct one, it is possible to use the averaged errors and scores. If the regulation chain involves genes *i*, *j* and *k*, with links *i*→*j*, *i*→*k* and *j*→*k*, the normalized sum of all models errors is

$$\epsilon_u = \frac{(e_{ij} + e_{ik}) - (e_{ij} + e_{jk})}{\max\{(e_{ij} + e_{jk}), (e_{ij} + e_{ik})\}}, \tag{3}$$

and for scores

$$\rho_u = \frac{(s_{ij} + s_{jk}) - (s_{ij} + s_{ik})}{\max\{(s_{ij} + s_{jk}), (s_{ij} + s_{ik})\}}. \tag{4}$$

Therefore, when the average between ϵ_u and ρ_u is greater than zero, it can be inferred that *i*→*j*→*k* has the best value, so that *i*→*k* must be removed; and when the average is less than zero, the best value is for *i*→*j* and *i*→*k* regulations, determining that *j*→*k* must be removed.

Considering again the example at Figure 4c), *B*→*E* is effectively removed from the matrix (cell highlighted in grey). The resulting matrix after the application of this rule is shown in Figure 4d).

Finally, at the end of the application of the three rules explained in 2.3.2, 2.3.3 and 2.3.1, a final GRN can be obtained from the resultant scoring matrix. For example, from Figure 4d) it is possible to obtain the GRN shown in Figure 5 by drawing the *regulator-regulated* relations represented by the matrix.

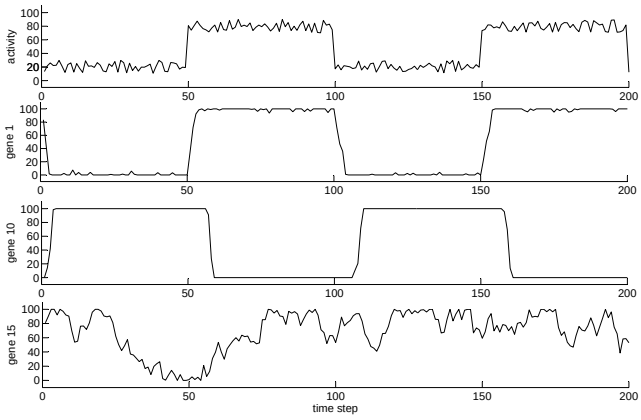

Fig. 6. Example of time series for the activation signal (top) and the expression of genes 1, 10 and 15.

3 MATERIALS

In this section an artificial dataset with known connections is described. A real dataset which have been used to validate the proposal is also presented.

3.1 Artificial data set

The artificial data set was developed by Smith et al. [22] and improved by Yu et al. [23]. They modeled a genetic regulatory network of arbitrary structure and measured values for gene expression levels at discrete time-steps. The method was the same used by Knott. et al [13] to model the gene expression values and validate its gene regulatory model.

Figure 6 shows examples of some of the time series used in this study. It can be seen that an increase in *activity* (top trace) is followed by upregulation of *gene 1* (second trace from top) with a slight time lag. When activity drops, then *gene 1* returns to hover near its target level, also with a little delay. *Gene 10* (third trace from the top) is down-regulated considerably later than the response of *gene 1* to activity, showing that the regulatory effects are propagated through a network over time. *Gene 15* (bottom trace) is an unregulated gene which changes in a wide range and without relationship to the activity or the other genes shown. The activity level acts as an activation signal starting randomly between a low or high level, and directly affecting the expression levels of the first two genes of the GRN. These two genes, in turn, affect the expression levels (time series) of eight downstream genes. For these time series, the corresponding network is shown in Figure 7. It contains 20 genes, where only 10 are associated among them through regulatory interactions, while the remaining 10 genes serve as distractors. Each connection between genes has a number that represents the expression proportion that is added or substracted from the regulator

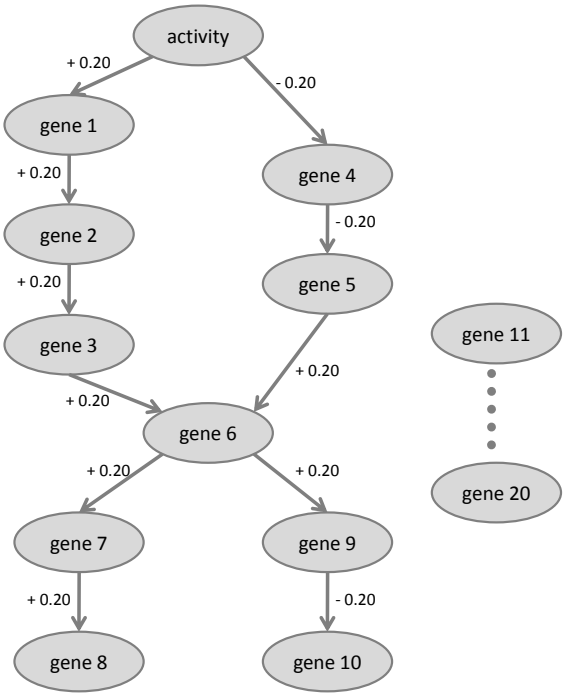

Fig. 7. Artificial data set GRN topology used as golden standard.

gene signal [22]. This proportion indicates whether the predecessor upregulates or downregulates the successor gene signal.

At each time point, the expression levels of genes in the network are governed by the expression levels of their regulators, a degradation factor and a noise factor. In contrast, the expression levels of the remaining genes randomly fluctuate within the upper and lower expression level bounds. Due to the stochastic nature of the noise in the system, the output will slightly differ in each replicate but it will be governed by the same underlying network interactions.

In this work 10 replicates will be used, where each one comprised 20 gene expression levels at 200 time points, sampled at five minutes intervals. This multiple replicates method was originally suggested in [24] for solving the dimensionality problem, by combining multiple microarray datasets from different conditions for inferring the GRN, in order to derive in the most consistent network structure with respect to all the datasets. Similarly, we used several replicates in order to train our models with enough samples from the same GRN. The 80% of the whole dataset will be used for training, while the remaining 20% for validation.

3.2 Five-gene network in *Yeast*

Yeast (*Saccharomyces cerevisiae*) cell cycle-regulated genes were identified by microarray hibridation [25]. The time series of gene-expression data from yeast protein synthesis[2] has 17 sampling points and the observation interval is 10 min. Data between timepoints have been normalized with respect to each other. In this work, data for five genes have been selected (HAP1, CYB2, CYC7, CYT1, COX5A) because the relations among those genes have been reported and validated by biological experiments [26].

4 Experimental Setup

The experimental setup will be explained here, detailing neural networks configuration. The performance measures for evaluating the proposed *GRNNminer* will be also presented.

4.1 Neural models configuration

A NN with temporal delays has a particular input layer, in which the number of elements depends on the number of temporal delays (τ) considered at the input data. In this work, two values were considered: τ = {2,3}. As regards hidden layer, a simple heuristic was adopted in which the number of hidden neurons were 50%, 100%, 150% and 200% of the inputs number. For example, for input layer with 4 elements (τ = 3), the hidden layer can have 2, 4, 6 or 8 neurons. The output layer has only one neuron, which represents the value of the target gene expression at time t.

Weights and biases were initialized according to the Nguyen-Widrow initialization algorithm [27], which chooses random values in order to distribute the active region of each neuron approximately evenly across the input space. The models were trained using a function that updates weights and bias according to Levenberg-Marquardt optimization [28]. The number of training epochs used as stop criteria were 50, 100 and 500. In this work, the parameter θ was tested between 0 and 1, with an incremental factor of 0.01.

4.2 Performance measures

Two measures were used in this work: accuracy and sensitivity. Accuracy is the proportion of true results (both true positives and true negatives) in the population, whereas sensitivity relates to the ability to identify positive results. The known GRN of the dataset (Figure 7) was used as the golden reference for performance calculation. *Accuracy* and *sensitivity* are defined as

$$A = \frac{TP + TN}{TP + TN + FP + FN}, \tag{5}$$

$$S = \frac{TP}{TP + FN}, \tag{6}$$

where TP, TN, FP and FN stand for true positives, true negatives, false positives and false negatives respectively.

2. http://www.wanghaixin.com/biowq2007.html

TABLE 2
Accuracy when the highest sensitivity are achieved in each experiment.

Window size (τ)	Epochs	θ	A [%]	S [%]
2	50	0.02	**99.71**	**100.00**
	100	0.02	99.12	88.89
	500	0.02	99.42	100.00
3	50	0.02	99.12	88.89
	100	0.02	98.83	77.78
	500	0.02	99.12	88.89

It is important to note that if accuracy is 1, all the relations are discovered and there are no false positives in the results. That is to say, when accuracy is 1, sensitivity is 1, but not the opposite. On the other hand, if sensitivity is 1, all the desired relations are discovered, but there can be some false positives present in the results.

5 Results and Discussion

The experimental results obtained on two GRN mining problems are shown in this section. First, the performance measures obtained from the results of mining the artificial dataset will be shown, in order to demonstrate the strengths of the proposed method. The discovered GRN will be analysed and a discussion with related work will be done. Second, the results from the application of the *GRNNminer* to a real dataset will be discussed.

Table 2 presents global results of the proposed mining approach on the artificial data set. The first column indicates the window sizes used, representing the temporal delay taken into account at the NN input. The different training epochs considered are shown at the second column. The third column presents the best θ for the threshold rule. The accuracy is presented at column four, while the sensitivity is shown in the last column.

Analyzing this table, two important results can be highlighted. First, by using *GRNNminer* it is possible to obtain all the relations of the golden reference, because the sensitivity was 100% in two cases (rows 1 and 3). Secondly, accuracies are higher than 99% for these two cases so that there are very few extra relations discovered. The best accuracy obtained was of 99.71% (row 1, shown in bold), which means that there is only one extra relation, and the proposed method is capable of discovering a GRN that is almost identical to the golden reference. That is to say, all of the gene-to-gene relations that have to be discovered were obtained, adding only one relation that is not present in the reference network. Sensitivity and accuracy show that it is possible to find a good model that takes into account the temporal structure of the data independently of the number of epochs. That is, the NN with temporal delays can identify the dynamics of the actual model, achieving very similar results with either high or low number of training epochs. Therefore, the neural models proposed here can effectively deal with the identification of two related genes by modeling their temporal profiles.

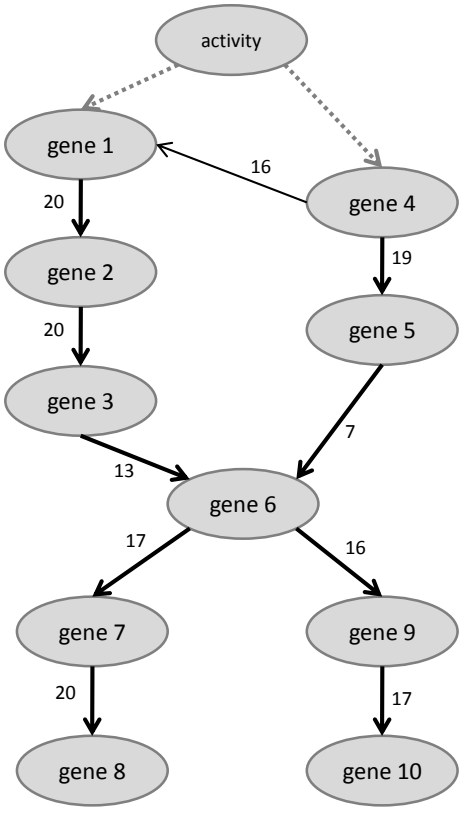

Fig. 8. Discovered GRN. Dotted lines indicate the activity influence over genes 1 and 4. Thick lines are the relations discovered by *GRNNminer* that match the golden reference relations in the dataset. Thin line is the FP that was discovered by the proposed method. Each line has it corresponding score.

TABLE 3
Comparison of GRNNminer against literature methods.

Methods	*A [%]*	*S [%]*
Pearson correlation	92.69	100.00
Rank correlation	83.92	100.00
Mutual Information	65.79	100.00
Knott et al. [13]	99.71	88.89
Smith et al. [22]	98.54	88.89
GRNNminer	**99.71**	**100.00**

From the score matrix it is possible to obtain the GRN shown in Figure 8. It can be clearly seen that *GRNNminer* was able to discover all the regulatory interactions existing between genes, as shown by thick lines. Only one extra relation was discovered, indicated with thin line (gene 4 $\rightarrow$ gene 1).

In this same problem, our proposal was compared against five methods. Results are shown in Table 3.

The first column shows the methods in comparison. The second and third columns report the accuracy, obtained when the highest sensitivity is achieved, respectively. The first row shows the results of a method based on Pearson correlation [29], where the correlation over all possible pairwise relationships between genes is calculated, and only those values that are higher than a threshold are selected as indicative of a possible regulation relationship between two genes. To determine this threshold, all possible values were evaluated between 0 and 1 (with step 0.01), selecting the one which allowed to find the best sensitivity and accuracy. Similarly, Rank Correlation [29] and Mutual Information [30] methods are reported in the second and third row, respectively.

In comparison to the method proposed in Knot et al. [13] for the same GRN, our proposal was able to discover all the interactions, while the Knot et al. method was able to infer only eight of the nine regulatory interactions. Their analysis was unable to identify gene 3 as a regulator of gene 6. Since it is was not specified in their work, we assumed for the best that they did not have any false positive. Another related work [22] that used the same data set and GRN was also unable to predict the regulation of gene 6 by gene 3, as well as five incorrect extra links between genes were additionally found. Its corresponding accuracy and sensitivity are reported in the fifth row. It should be highlighted here that the GRNNminer presented in this work, instead, was capable of finding this important regulation between genes 3 and 6, with the best accuracy and sensitivity (as shown in the sixth row).

Analyzing Table 3 in detail it can be stated that, although by using the proposals of Knott et al. and Smith et al. it is possible to obtain most part of the reference network (accuracies of 99.71% and 98.54%, respectively), it is not possible to fully reconstruct it, since the sensitivity was 88.89% in both cases. As regards Rank correlation and Mutual Information methods, in spite they can obtain the complete reference network there are also so many false positives that impact negatively on the performance of both methods (accuracies of 83.92% and 65.79%, respectively). By using the Pearson correlation, high sensitivity and accuracy (92.69%) can be obtained. However, GRNNminer has a better accuracy (99.71%), which means that it not only obtains the reference network, but also obtains only one false positive. This better performance of GRNNminer is due to the advantage of modeling the dynamic relationship between genes profiles. Unlike the traditional correlations methods, the temporal dynamic is considered explicitly in GRNNminer by modeling gene regulation through neural networks with temporal delays at the input.

In order to further evaluate the scalability and robustness of GRNNminer, this experimental work was extended to discover the same GRN with 100 genes (9900 putative relationships between genes). The accuracy increased to 99.99% at a sensitivity of 100%. It shows the capability of the proposed method for mining GRNs from larger datasets.

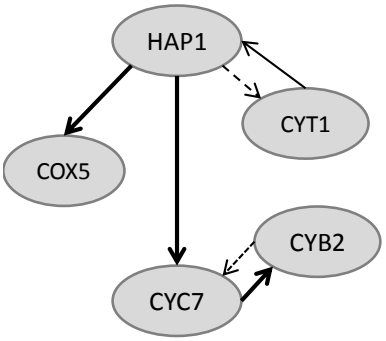

Fig. 9. Discovered yeast GRN. Thick lines are the relations discovered by *GRNNminer* that matched the golden reference relations in the yeast dataset. Thin line is the relation discovered that is not present in the subjacent network. Dashed lines are the relations that the proposal was not able to discover because discards mutual regulations. Each line has it corresponding score.

GRNNminer method was also tested with the real dataset described in Section 3.2. Figure 9 shows the GRN reconstruction after the application of the mining method. In this case, three correct relations have been found: HAP1→COX5, HAP1→CYT1 and CYC7→CYB2 (thin lines). All the relations discovered by the proposed method are real regulations between genes in the yeast protein synthesis pathway. These results are in agreement with the experimental findings in [26]. Since mutual regulation is not considered in GRNNminer, the CYB2→CYC7 relation has not been found (dotted line). Another case to be analyzed is the false positive between HAP1 and CYT1. Although the real sense of the relation between CYT1 and HAP1 is the opposite, it should be highlighted that the relation was effectively found by the proposed approach. The same problem was addressed in [31], where the same relations between genes have been discovered, plus one extra regulation between CYB2→CYC7 and another one between HAP1→CYT1. That work has also found an extra relation COX5→HAP1 that is not actually present in the real network. It is worth noting that this false positive has not been indicated by GRNNminer due to the fact that mutual regulation is not considered into the mining rules and our approach based on the minimum error of the neural models has correctly identified the regulation HAP1→COX5 as stronger (it has obtained a higher score) than COX5→HAP1.

6 Conclusions and Future Work.

A novel approach for successfully obtaining a GRN from genes expression data was proposed and explained in detail. *GRNNminer* involves modeling each pair of genes interaction with NNs. Specifically, a pool of multilayer perceptrons was used to model all the possible combinations of gene-to-gene relations between genes present in the dataset. The ability of modeling each relation was measured according to the generalization error. A scoring method was proposed in order to determine the most probable interactions in the dataset. From the resulting scoring matrix, a set of mining rules were applied to discover the underlying GRN. Several experiments were done over a known GRN present in an artificial dataset, in comparison against traditional and recent methods in literature. By calculating sensitivity and accuracy measures, experimental results demonstrate that the proposal is capable of discovering all the gene-to-gene relations and reconstruct the subjacent GRN. Moreover, applying the same method over a real dataset, it was possible to obtain the interactions between real genes. The capability of our mining approach to identify the potential existing relations between genes could be very useful because it allows to focus the attention over a set of genes. *GRNNminer* could be used as a starting point from where researchers can reconstruct or build a regulation pathway, later to be tested or validated through wet experiments.

Future work involves the scaling up of the proposed method to more complex problems, including a large number of genes. Also, in order to improve the estimation of the temporal delays to be considered at the input of each neural model, we will study the automatic evaluation of the temporal dynamics at each gene expression data.

References

[1] M. Hecker, S. Lambeck, S. Toepfer, E. van Someren, and R. Guthke, "Gene regulatory network inference: Data integration in dynamic modelsA review," *Biosystems*, vol. 96, no. 1, pp. 86–103, 2009.

[2] A. J. Hartemink, "Reverse engineering gene regulatory networks," *Nat Biotech*, vol. 23, no. 5, pp. 554–555, May 2005.

[3] R.-S. Wang, Y. Wang, X.-S. Zhang, and L. Chen, "Inferring transcriptional regulatory networks from high-throughput data," *Bioinformatics*, vol. 23, no. 22, pp. 3056–3064, Nov. 2007.

[4] H. D. Jong, "Modeling and simulation of genetic regulatory systems: A literature review," *Journal of Computational Biology*, vol. 9, pp. 67–103, 2002.

[5] M. P. Styczynski and G. Stephanopoulos, "Overview of computational methods for the inference of gene regulatory networks," *Computers & Chemical Engineering*, vol. 29, no. 3, pp. 519 – 534, 2005.

[6] S. Mitra, R. Das, and Y. Hayashi, "Genetic networks and soft computing," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 1, pp. 94–107, 2011.

[7] D. Muraro, U. Voss, M. Wilson, M. Bennett, H. Byrne, I. D. Smet, C. Hodgman, and J. King, "Inference of the genetic network regulating lateral root initiation in arabidopsis thaliana," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 1, pp. 50–60, 2013.

[8] S. Bornholdt, "Boolean network models of cellular regulation: prospects and limitations." *J R Soc Interface*, vol. 5 Suppl 1, 2008.

[9] C. H. A. Higa, T. P. Andrade, and R. F. Hashimoto, "Growing seed genes from time series data and thresholded boolean networks with perturbation," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 1, pp. 37–49, 2013.

[10] A. Werhli and D. Husmeier, "Reconstructing Gene Regulatory Networks with Bayesian Networks by Combining Expression Data with Multiple Sources of Prior Knowledge," *Statistical Applications in Genetics and Molecular Biology*, vol. 6, no. 1, pp. 1–45, 2007.

[11] D. Husmeier, "Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks," *Bioinformatics*, vol. 19, no. 17, pp. 2271–2282, 2003.

[12] S. Ando and H. Iba, "Inference of gene regulatory model by genetic algorithms," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 1, 2001, pp. 712–719 vol. 1.

[13] S. Knott, S. Mostafavi, and P. Mousavi, "A neural network based modeling and validation approach for identifying gene regulatory networks," *Neurocomputing*, vol. 73, no. 13-15, pp. 2419–2429, 2010.

[14] H. Hache, H. Lehrach, and R. Herwig, "Reverse engineering of gene regulatory networks: A comparative study," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2009, no. 1, p. 617281, 2009.

[15] H. L. H. Hache, C. Wierling and R. Herwig, "Reconstruction and validation of gene regulatory networks with neural networks," in *Proceedings of the 2nd Foundations of Systems Biology in Engineering Conference (FOSBE 07)*, 2007, pp. 319–3241.

[16] A. Chowdhury, M. Chetty, and N. Vinh, "Incorporating time-delays in s-system model for reverse engineering genetic networks," *BMC Bioinformatics*, vol. 14, no. 1, p. 196, 2013.

[17] N. A. Barker, C. J. Myers, and H. Kuwahara, "Learning genetic regulatory network connectivity from time series data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 1, pp. 152–165, 2011.

[18] G. Stegmayer and O. Chiotti, "Volterra NN-based behavioral model for new wireless communications devices," *Neural Computing and Applications*, vol. 18, pp. 283–291, 2009.

[19] D. Chaturvedi, *Soft Computing Techniques - Studies in Computational Intelligence*. Springer-Verlag, London, 2008.

[20] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, 1999.

[21] D. Rumelhart, G. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 1, pp. 533–536, 1986.

[22] V. A. Smith, E. D. Jarvis, and A. J. Hartemink, "Evaluating functional network inference using simulations of complex biological systems," *Bioinformatics*, vol. 18, no. 1, pp. S216–S224, 2002.

[23] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis, "Advances to bayesian network inference for generating causal networks from observational biological data," *Bioinformatics*, vol. 20, no. 18, pp. 3594–3603, 2004.

[24] Y. Wang, T. Joshi, X.-S. Zhang, D. Xu, and L. Chen, "Inferring gene regulatory networks from multiple microarray datasets," *Bioinformatics*, vol. 22, no. 19, pp. 2413–2420, 2006.

[25] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher, "Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization," *Mol Biol Cell*, vol. 9, no. 12, pp. 3273–3297, Dec. 1998.

[26] J. C. Schneider and L. Guarente, "Regulation of the yeast cyt1 gene encoding cytochrome c1 by hap1 and hap2/3/4." *Molecular and cellular biology*, vol. 11, no. 10, pp. 4934–4942, 1991.

[27] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights." in *Proceedings of the International Joint Conference on Neural Networks*, 1990, pp. 21–26.

[28] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.
[29] J. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference, Fourth Edition: Revised and Expanded*. Taylor & Francis, 2014.
[30] T. Cover and J. Thomas, *Elements of information theory*. Wiley, 1991.
[31] B. Yang, Y. Chen, and M. Jiang, "Reverse engineering of gene regulatory networks using flexible neural tree models," *Neurocomput.*, vol. 99, pp. 458–466, Jan. 2013.

MARIANO RUBIOLO received the Engineering degree in Information Systems from Universidad Tecnologica Nacional Facultad Regional Santa Fe, Argentina, in 2007, and the Ph.D. degree in Engineering in Information Systems from Universidad Tecnologica Nacional Facultad Regional Santa Fe, Argentina, in 2014. He was Auxiliar Professor at the Department of Information System Engineering in UTN-FRSF from 2008 and 2014. Recently, he was named Full Adjunct Professor in the same department. He is currently Research Assistant at the National Scientific and Technical Research Council (CONICET) Argentina. His current research interests are in the fields of computational intelligence, data mining and bioinformatics.

DIEGO H. MILONE received the Bioengineering degree (Hons.) from National University of Entre Rios (UNER), Argentina, in 1998, and the Ph.D. degree in Microelectronics and Computer Architectures from Granada University, Spain, in 2003. He was with the Department of Bioengineering and the Department of Mathematics and Informatics at UNER from 1995 to 2002. Since 2003 he is Full Professor in the Department of Informatics at National University of Litoral (UNL). Since 2006 he is a Research Scientist at the National Scientific and Technical Research Council (CONICET). From 2009 to 2011 was Director of the Department of Informatics and from 2010 to 2014 was Assistant Dean for Science and Technology. His research interests include statistical learning, pattern recognition, signal processing, neural and evolutionary computing, with applications to speech recognition, computer vision, biomedical signals and bioinformatics.

GEORGINA STEGMAYER received the Engineering degree in Information Systems from Universidad Tecnologica Nacional Facultad Regional Santa Fe, Argentina, in 2000, and the Ph.D. degree in Electronic Devices from Politecnico di Torino, Italy, in 2006. Since 2007 she is Professor at the Department of Informatics in UTN-FRSF and UNL. She is currently Adjunct Researcher at the National Scientific and Technical Research Council (CONICET) Argentina. She is author and co-author of numerous papers on journals, book chapters and conference proceedings in artificial neural networks. Her current research interests are in the fields of data mining and pattern recognition with application to bioinformatics.