Gene Expression Analysis using Markov Chains extracted from Recurrent Neural Networks

Ígor Lorenzato Almeida[1], **Denise Regina Pechmann**[1], **Maicon de Brito do Amarante**[2], **Adelmo Luis Cechin**[3]

[1]Instituto Federal Farroupilha – Campus Santo Augusto
Santo Augusto – RS – Brazil

[2]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

[3]Mestrado em Computação Aplicada
Universidade do Vale do Rio dos Sinos – São Leopoldo, RS – Brazil

{lorenzato,denise.r.p,adelmo.cechin}@gmail.com, mbamarante@inf.ufrgs.br

Abstract. *This paper presents a new approach for the analysis of microarray data by the use of Recurrent Neural Networks (RNNs) as a time model of the gene regulatory network. Our method extracts a Markov Chain (MC) from a trained RNN and the relations among genes in each MC state. We propose to use the learning ability of RNNs for the automatic construction of the model with the gene interactions represented by the weights and afterwards to use an algorithm to extract these relations in the form of MCs and linear matrices easily visualized in the form of graphs of states and genes. The graph of states show the evolution of the gene expression levels in time while the gene graph shows the dependencies among genes in each Markov state.*

1. Introduction

With the increasing advance of genomic projects, a great amount of microarray data has been yielded. These high troughput techniques have made necessary the development of automatic analysis programs of these generated data, which is shifting from the common *ad hoc* analysis to statistical analysis.

The microarrays were developed on the 90's and it is possible today to analyse the gene expression of thousand of genes at the same time. Using them, it is possible to study the gene expression patterns, that are the base of the celular fisiology, analysing the activation (or inactivation) of the genes in a certain enviroment[Alberts et al. 2004],[Causton et al. 2003],[Kohane et al. 2003]. This work intends not only to analyse which genes are, or not, being activated but also the interactions among them.

Microarrays contain the gene expression levels of many genes simultaneously (for instance, 30000 genes) at different time steps. At each time step, the expression level of a certain gene depends on the actual expression of all the other genes and also on their past values. This net of interactions cannot be represented by a single net of linear relations, because some genes may influence positively or negatively other genes depending on their absolute values. Therefore, only by using a net of nonlinear relations, such as those present in Recurrent Neural Networks, it is possible to capture the relations among gene expression levels. Recurrent Neural Networks may be automatically trained with time series. At the same time, these remove the noise interactions, remaining only the main relations among variables in the network structure. However, the knowledge in the form of weights is difficult to understand. Certainly, if we are interested only in the prediction of the gene expression levels, just to train the network would suffice. However, most users are interested in understanding and visualizing the network of gene interactions. Then, in this work, we compute a Markov Chain from the Recurrent Neural Networks[Pechmann and Cechin 2005],[Pechmann and Cechin 2004]. Once learned in the weights of the neural network, these temporal relations may be extracted in the form of a Markov Chain states, which is a form easily understandable by any scientist.

The states and transitions in the Markov Chain represent the nonlinear temporal relations among discretized gene expression levels. Our extraction method guarantees that in each state, only linear relations remain and these are represented in the form of a gene graph. Then if one gene is influencing positively other genes and afterwards this influence turns to be negative, then we extract two markov states, one for each linear influence. This linear relations among genes is of great value for biologists promoting a better understanding about the organism under analysis. First, by understanding the network of gene activation, it is possible to predict the reaction of the organism (activated genes) under the influence of drugs, environment conditions or phase of the life cycle. Second, the temporal relation in the form of a Markov Chain allows the scientist to understand and predict under which conditions the organism changes its linear network of gene expression levels to another one, and therefore how the organism adapts its network to adverse environmental conditions, for instance, availability of different elements, food, minerals, temperature and stress conditions. Different conditions need different linear influences among genes, even to the degree that a positive influence between two genes must turn to a negative influence. Extracting this information without the use of computational tecniques is impracticable because of the great amount of data and genes. Our work presents a first approach to solve the automatic extraction of such information and its application to the *Stanford Microarray Database* (SMD) [Ball et al. 2005].

Thus, this work presents in Section 2 a brief description about Markov Chains. For a good introduction to Recurrent Neural Networks, see [Haykin 1999]. The extraction methodology is described in Section 3. In section 4, we describe the data set used, as well as the data pre-processing and the results obtained. Finally, in Section 5, the conclusions are presented.

2. Markov Chains

A stochastic process is a colection of random variables indexed by a time parameter n, defined in a space named state space. The value x_n assumed by the random variate X_n in a certain instant of time is called state. A random process, X_n, is a Markov process if the future of the process, gived the present, is independent of the past of it.

In a Markov Chain of order 1, the actual state depends only of the previous state, as expressed in Eq 1.

$$P[X_{n+1}|X_n = x_n, ..., X_1 = x_1] = P[X_{n+1} = x_{n+1}|X_n = x_n] \tag{1}$$

So, a sequence of random variables $X_1, X_2, ..., X_n, X_{n+1}$ forms a Markov Chain if the probability of the system to be in the state x_{n+1} in time $n+1$ depends exclusively of the probability that the system was on state x_n in time n [Haykin 1999].

In a Markov Chain, the transition of a state to another is stochastic, but the production of an output simbol is deterministic. The probability of transition of a state i in time n to the state j in time $n+1$ is given by

$$P[X_{ij}] = P[X_{n+1} = j | X_n = i] \tag{2}$$

All the transition probabilities must satisfy the following conditions:

$$p_{ij} \geq 0 \text{ for all } (i,j)$$

and

$$\sum_j p_{ij} = 1 \text{ for all } i.$$

In the case of a system with a finite number K of possible states, the transition probabilities constitute a matrix K-by-K, called stochastic matrix, whose individual elements satisfy the described conditions in Eqs 1 and 2, where the sum of each line of the matrix should result in 1 [Manning and Schutze 2000].

3. Extracting of Markov Chains from RNNs

In this section, we presented the knowledge extraction method described in [Pechmann and Cechin 2005] and [Pechmann and Cechin 2004]. The objective of the knowledge extraction is to generate a concise and easily understandable symbolic description of the knowledge stored in the recurrent model [Cloete and Zurada 2000],[Giles et al. 2001],[Andrews et al. 1995].

Recurrent Neural Networks are nonlinear dynamic systems sensible to initial conditions and they are able to store the dynamics of the gene expression levels in the form of parameters.

The state extraction is related to the division of the input space or of the related neural space (space spanned by the neuron activations) of the RNN, or to clustering methods. The clustering method investigated is the fuzzy clustering. The main objective here is to find a state (discrete) representation of the hidden layer activation. This state representation is composed of a set of membership functions, which may be reduced to discrete sets if we define a threshold, for instance 50% as the border definition of each state, and a linear approximation relating input and outputs. This linear approximation is valid only in the respective state. Thus, the membership value can be interpreted as a measure of the validity of the linear approximation. To combine the individual neuron membership functions in a membership function for the network, we use the operators of the Fuzzy Logic. If we know (certainty indicated by the membership value) where each hidden neuron is working, then the whole network can be collapsed into a linear relation among inputs and outputs. Therefore, each state is represented by such a linear relation and a combination of fuzzy sets for the hidden neurons.

Two compromises have to be reached in the choice of the number of membership functions used to represent the activation of each neuron. Normally, larger states contain more data and represents larger regions of the neural work space and input spaces, resulting in a more spare representation, which is also easier to understand but represents the data in a less exact way. Contrary to this, smaller more numerous neuron states represent few data very exactly, but many of them are required to represent the whole input space. They are more difficult to understand and to analyze. If we choose a too fine representation for each neuron, many resulting states will contain few data or even no data [Cloete and Zurada 2000].

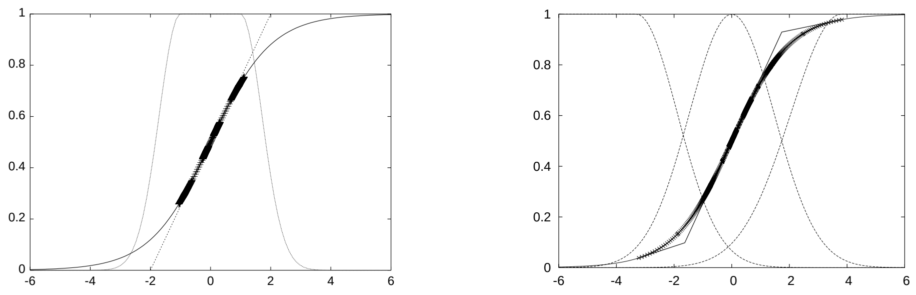


Figure 1. Activations for two neurons in the hidden layer during the simulations. The work regions of the activation function of the nonlinear neurons is shown with crosses. Neurons that work only in the central linear region of the activation function occupy only one membership function (left). Neurons that work along a larger part of the activation function, that is, over several linear regions of the activation function, require more fuzzy sets (right).

To compute the adequate number of membership functions for each neuron, the following procedure is applied. After the training and test of the RNN, the data is presented again to the network to compute the statistical distribution of the activations of the neurons in the hidden layer. The construction of the membership functions is based on the Gauss function considering the mean and standard deviation of the activation values for each neuron in the hidden layer (see Figure 1). According to the occupancy of the neural space, the number and form of the linear approximations are chosen. The choice of the number of fuzzy sets for each neuron is based on the error of the piecewise linear approximation in the work regions of the hidden neurons.

The Figure 1 shows the membership functions for two hidden neurons and the correspondent linear relations: $0.25x + 0.5$ for the first neuron and three linear functions: $0.034x + 0.14$, $0.25x + 0.5$ and $0.021x + 0.89$ for the second one. This way, our approach does not attribute states to a neuron if this requires just one to be represented.

To obtain the states for the whole network, we proceed in the following way. If μ_{11} is the membership function of the first neuron and μ_{12}, μ_{22} and μ_{23} of second neuron, then the combination of them gets the membership for the whole network. The combination is implemented with the *product* or *min* operator of the Fuzzy Logic, for example, $\mu_{11}\mu_{22}$. Because in each neuron state, we may represent the neuron by its linear approximation ($0.034x + 0.14$, for instance), the combination of the membership functions of each neuron allows us to collapse the whole network into a single linear relation among network input and outpus. Unfortunately, this makes the number of fuzzy sets for the whole network $O(p^h)$, where p is the number of fuzzy sets used for each hidden neuron and h is the number of hidden neurons. Then we perform a statistical analysis of the occupancy of each whole network state to discard those less occupied. Therefore, the user may choose between an investigation about typical states during the experiment or to investigate rare states with uncommon relations among genes that appear due to special conditions occurred during the experiment. Thus, the obtained number of states reflects the richness or variability of the possible interactions among genes.

The state transitions are detected through the time series used in the clustering. It is performed using the clusters as states and identifying each state transition. To obtain an approximation of the probability values, we compute the transition frequencies.

4. Experiments

Experiments were carried out using a set of 80 gene expression vectors for 2467 yeast genes that were selected by Eisen *et al* [Eisen et al. 1998] and used with succes by Brown *et al* [Brown et al. 2000]. The data were generated from spotted arrays using samples collected at various time points during the diauxic shift (transition from anaerobic fermentation of glucose to aerobic respiration of ethanol), the mitotic cell division cycle, sporulation, and temperature and reducing shocks. This data is part of the *Stanford Microarray Database* (SMD) [Ball et al. 2005] and are available at *http://rana.stanford.edu/clustering*.

For the Markov Chain generation, first the data must be used to train a RNN. The great amount of patterns in the data set, its high level of correlation and noise makes this data inadequate for direct network training. Then, we applied the described pre-processing before submitting the network to training.

4.1. Pre-processing

First, we reduced the data set to data set representatives. This breaks the statistical dependency of our network on the more common genes and on the less representative ones. Many similar gene expression patterns are reduce to few ones. This maintains the diversity and richness of the data without loosing information and causing the network to represent only the maybe not so interesting bulk of genes. Such caracteristics were unknown at this stage of our investigation. Thus, aiming to discover the more proeminent patterns in this data and to determine how many they are, the patterns were processed with Self-Organizing Maps [Kohonen 1997]. This way, we get a balanced training set appropriate for the Markov Chain extraction.

The number of training features or gene representatives for the RNN should not be so many making the training process impracticable, and neither so small that could not represent all the information present in the database.

The features can be determined by the analysis of the SOM trained with the whole data [Almeida et al. 2006]. The SOM has the ability to divide the different features present in the set. After the training, we count the number of data examples represented by each neuron and a threshold is chosen. This threshold will determine the minimum amount of genes that each SOM neuron must have to be used as RNN input. In this experiment, this threshold was 35 gene expressions. Therefore, each RNN input represents at least 35 genes with similar time behaviour. Any relation among two different input and outputs detected in the extraction represents a relation between two groups of genes.

Certainly it is not possible to detect which gene is causing those transitions in the others because those genes have the same time series and it is not possible, based only on the available data, to differentiate among them. Using the threshold of 35 genes, we obtained 11 features (each feature represents the time pattern of at least 35 genes) which are used as RNN network inputs. The choice of 35 genes is based on the histogram of the number of genes represented by each SOM neuron (see Figure 2).

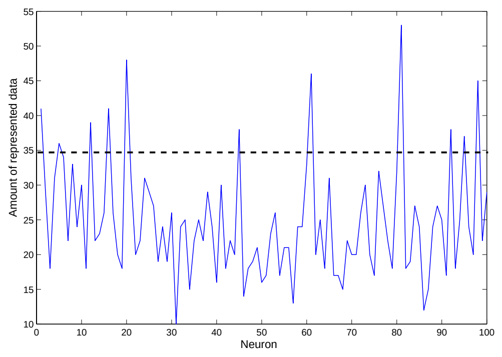


Figure 2. Amount of data represented for each neuron of SOM.

Once those features were determined and identified we have chosen only 36 gene expressions for each of the 11 features. Some of these features have more than 36 gene expressions, but we choose only the 36 most similar to the SOM codebook vector. This way, the overrepresented features in the data set are balanced with those with less patterns.

4.2. RNN Training

To obtain a good model for the knowledge extraction phase, several network topologies, all based on the Jordan Architecture, were trained and tested. The training database was used with networks consisting of three layers, differing by the number of neurons in the hidden layer. Networks with 1, 3, 5, 10, 15 and 20 hidden neurons were tested with the RMS error, as can be seen in the Table 1.

Table 1. Comparation among RNN Architectures.

Number of hidden neurons	RMS error
1	4.135321
3	3.956288
5	3.912200
10	3.866480
15	3.771372
20	3.733685

The network with 5 neurons in the hidden layer was chosen (see Figure 3). This was based on the good validation results presented by the network and on the small number of hidden neurons, what enables the extraction of a formal model of representation with a high level of comprehensibility.

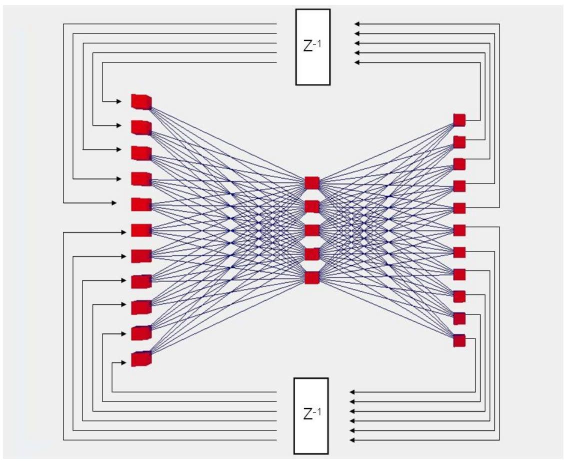


Figure 3. RNN based on the Jordan Network's Architecture and used in the Markov Chain extraction.

4.3. Markov Chain Extraction

For the extraction of the Markov Chain, 3 membership functions were built for each neuron, and by combining these functions, we could identify the membership functions for each state of the Markov Chain. From all possible Markov states and corresponding membership function, 24 were used to represent the training data.

The definition of the number of states was performed with the construction of a histogram, for which all membership functions were computed using all the training data. The 5 begining states represents 95% of the sampled data, and were used to compute the Markov Chain.

Then, according to the determined states, the activations of the hidden neurons were used by the fuzzy clustering process to determine the transition probability among states. This probability is computed by the statistical analysis of the time series, where the change in the state is detected and also its frequency. Therefore, as the states are already determined and the transitions among them computed, we obtain the Markov Chain representing the database (Fig. 4).

In this Markov Chain, which is represented in the form of a graph, the nodes are the states and the arcs are the probability of transition among them. The first and second states, for instance, represent 90% of the data. The probability of 86% means that, once the gene network of *Saccharomyces cerevisiae* is in the first state, which is defined by a certain interaction among genes, it remains there most of time. A Markov state is not necessarily a set of similar gene expression levels, but is characterized by influences among genes. The state is defined not by the expression level, but by the linear influence of one gene on the others. Since the data is a time series of continuous values, the actual gene expression level jumps smoothly from one state to the other. Then, the representation of this transition as a Markov Chain is a simplification of the real biological transition, because this transition occurs smoothly. However, the fuzzy membership functions give us this information and we are able by following their values, to know if we are entering or leaving one state.

Beyond this fuzzy concept of membership in each state, we have determined for

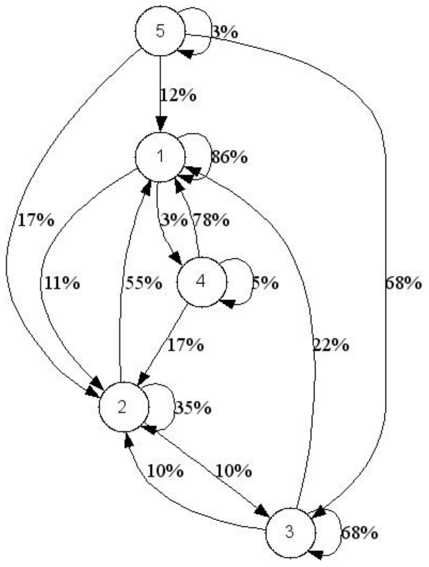


Figure 4. Markov Chain extracted from RNN with 5 hidden neurons, trained with microarray data.

each state the set of linear equations (influences among genes), that describe the interaction among genes. These are usually known as the gene regulatory network with the generalization, that now we are able to attribute if the influence is positive, negative or even zero. Further, by the membership value we may verify the validity of the above influences on time, that is, during which period of time of the experiment the regulatory network is valid. The Fig. 6 represents the two main Markov states (regulatory networks) and Fig. 5 the period of time in which the corresponding regulatory network is valid. The first regulatory network is valid in the first 60 time steps while the second one is valid in the last 20 time steps. We can see the different influences among the gene groups (nodes) and the influences (arcs) both in the form of activation (positive arcs) or inhibition (negative arcs). This figure describes a valid regulatory network among these gene groups in a certain period of time of the temporal series.

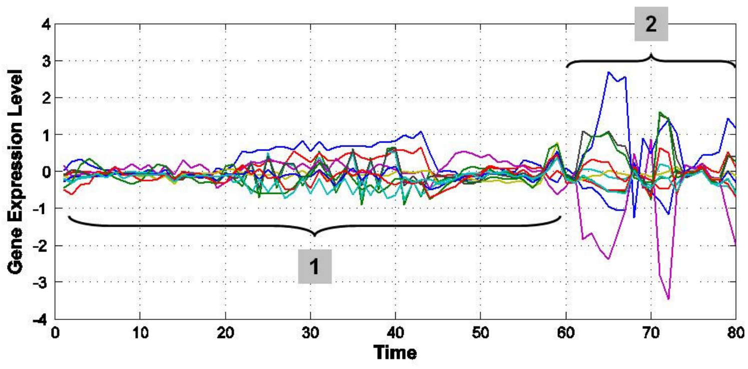
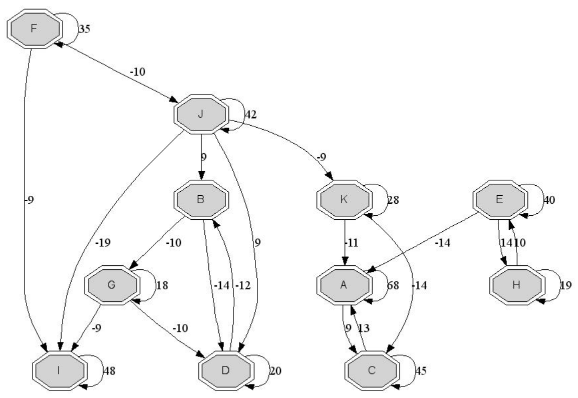
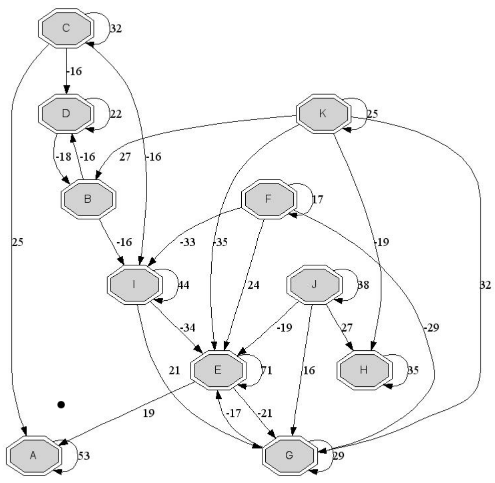


Figure 5. Period of time where each Markov state.

For example, gene group A activates and is activated by gene group C ($+9$ and $+13$) and is deactivated by K (-11) and L (-14) in the first state (first 60 time steps).



(a) Regulatort network corresponding to Markov state 1 in Fig. 4.



(b) Regulatory network corresponding to Markov state 2 in Fig. 4.

Figure 6. Graph representing the two main states of the Markov Chain extracted. The nodes represents the genes and arcs the influences.

However, as the organism enters the last 20 time steps, the influence of C on A increases $(+25)$, all other influences disappear and appears the influence of F on A, not directly present in state 1. In state 1, of all influences in the regulatory network, the greatest one is represented by J $\xrightarrow{-19}$ I (negative) while this influence disappears in state 2.

5. Conclusions

The analysis of genic expression levels with microarray techniques is becoming more and more common and it is also producing a huge amount of data. One of the problems encountered in this area is how to obtain useful information from time series.

Two aspects of interest for the researcher are the time evolution of the genic expression and their mutual influence in a regulatory network form. Thus, this work presents a original methodology for knowledge extraction from microarray data. We have showen the necessary data manipulation and pre-processing, training of the Recurrent Neural Network (time model) and extraction of information in the form of a Markov Chain and extraction of regulatory networks.

The Markov Chain represents the temporal relations among genes and expresses the transition probabilities. Membership values can be used to determine the exact time validity of each Markov state and the interaction graph, one for each Markov State, represents the influence of the one gene or another or the regulatory network. These relations are shown in the form of graphs.

These relations among genes are important in diverse areas, like the drug industry, which, with this information can develop strategies to reduce side effects in the cure of an illness. The automatic determination of the relations among gene expression levels is the subject of a lot of research to be done.

References

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. (2004). *Biologia Molecular da Célula*. Artmed, 4 edition.

Almeida, I. L., Pechmann, D. R., and Cechim, A. L. (2006). Markov chain inference from microarray data. *5th Mexican International Conference on Artificial Intelligence*, pages 133–141.

Andrews, R., Dietrich, J., and Tickle, A. B. (1995). A survey and critique of techniques for extracting rules from trained artificial neural networks. Technical report, Neurocomputing Research Centre, Australia.

Ball, C., Awad, I., Demeter, J., Gollub, J., Hebert, J., Hernandez-Boussard, T., Jin, H., Matese, J., Nitzberg, M., Wymore, F., et al. (2005). *Nucleic acids research*, 33(Database Issue):D580.

Brown, M. P. S., Brundy, W. N., Lin, D., Cristianini, N., Sugnet, C. W., Furey, T. S., Ares, M., and Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. *PNAS*, 97 - Part 1:262–267.

Causton, H. C., Quackenbush, J., and Brazma, A. (2003). *Microarray. Gene Expression Data Analysis, A Beginner's Guide*. Blackwell Publishing.

Cloete, I. and Zurada, J. M. (2000). *Knowledge-Based Neurocomputing*. MIT Press.

Eisen, M., Spellman, P., Brown, P., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868.

Giles, C. L., Lawrence, S., and Tsoi, A. C. (2001). Noisy time series prediction using recurrent neural networks and grammatical inference. In *Machine Learning*, volume 44(1-2), pages 161–183. Springer Netherlands.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice-Hall.

Kohane, I., Kho, A., and Butte, A. (2003). *Microarrays for an integrative genomics*. MIT Press.

Kohonen, T. (1997). *Self-organizing maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Manning, C. D. and Schutze, H. (2000). *Foundations of Statistical Natural Language Proc.* MIT Press.

Pechmann, D. R. and Cechin, A. L. (2004). Representação do comportamento temporal de redes neurais recorrentes em cadeias de markov. In *VIII Brazilian Symposium on Neural Neural Networks (SBRN2004)*, volume 1, pages 1–10.

Pechmann, D. R. and Cechin, A. L. (2005). Comparison of deterministic and fuzzy finite automata extraction methods from jordan networks. In *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, pages 437–444.