Gene Expression Analysis using Markov Chains extracted from RNNs

Ígor Lorenzato Almeida, Denise Regina Pechmann, Adelmo Luis Cechin

[1]Universidade do Vale do Rio dos Sinos - UNISINOS
Programa Interdisciplinar de Pós Graduação em Computação Aplicada - PIPCA
Av. Unisinos, 950 - Bairro Cristo Rei - CEP 93.022-000 São Leopoldo, RS - Brasil

{lorenzato, denise.r.p}@gmail.com, acechin@unisinos.br

Abstract. *This paper present a new approach for the analysis of gene expression, by extracting a Markov Chain from trained Recurrent Neural Networks (RNNs). A lot of microarray data is being generated, since array technologies have been widely used to monitor simultaneously the expression pattern of thousands of genes. Microarray data is highly specialized, involves several variables in which are complex to express and analyze. The challenge is to discover how to extract useful information from these data sets. So this work proposes the use of RNNs for data modeling, due to their ability to learn complex temporal non-linear data. Once a model is obtained for the data, it is possible to extract the acquired knowledge and to represent it through Markov Chains model. Markov Chains are easily visualized in the form of states graphs, which show the influences among the gene expression levels and their changes in time.*

1. Introduction

With the increasing advance of the genomic projects, a great amount of microarray data has been generated. These advances become the search for efficient computational techniques to data analysis really necessary.

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[1],[2],[3]. This work intends not only to analyse which genes are or not being activated but also the interaction that could exist among them.

Microarrays contain the gene expression levels of many genes simultaneously 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. 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, this would suffice, but most users are interested in understanding and visualizing the network of gene interactions. 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. Then, in this work, we compute a Markov Chain from the Recurrent Neural Networks[4],[5]. The temporal relation among the gene expression levels will be expressed in the form of a Markov Chain extracted for a RNN trained with microarrays data.

Although there exists many public database containing time series of gene expression levels, usually they do not have more than 80 time sample[6], what makes the acquisition of a good model a difficult task. Small time series may not be enough to extract all the relation existing in the data, so it was decided to work with data extracted from the *Stanford Microarray Database* (SMD) because it has one of the most longer time series of gene expression leves and also due to the fact that other authors have currently used the same data[7],[8],making possible a comparison of results.

Microarray data have tipically two drawbacks. First, it has a lot of noise [2], [3], [9], [10]. Second, it contains a lot of redundant data with correlated gene expression levels. As this data will be used as the input parameter of a RNN, it makes a dimensional and noise reduction necessary, in order to obtain useful results in the process of Markov Chain extraction [11], [12]. To guarantee good results a lot of tecniques have been used with success [6]. Among them, there are Hierarchical Clustering [10], Nearest Neighbours [13] and Self-organizing Maps (SOM) [14], which will be used in this paper.

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. 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, and therefore how it adapts to adverse conditions. Different environment conditions need different linear influences among genes, even to the degree that a positive influence between two genes needs to 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) [15].

Thus, this work presents in Section 2 a brief description about Markov Chains. The extraction methodology is described in Section 3. In section 4, we describe the experiment, the data 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.

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

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 [16].

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

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

All the transition probabilities must satisfy the following conditions:

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

and

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

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 [17].

3. Extracting Markov Chains from RNNs

In this section, it is presented the knowledge extraction method described in [4] and [5]. The objective of the knowledge extraction is to generate a concise and easily understandable symbolic description of the knowledge stored in the recurrent model [18],[19],[20].

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 work space of the RNN, or to clustering methods. The clustering method investigated is the fuzzy clustering. The main objective here is to find a state representation of the hidden layer activation. This state representation is composed of a set of membership functions and piecewise linear approximations. The membership value can be interpreted as a measure of the validity of the linear approximation. If we is known (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 relation and a combination of fuzzy sets for the hidden neurons.

Two compromises have to be reached in the choice of the number of states. 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. On contrary, smaller states, represent few data, very exact, but many of them are required to represent the same input space. They are more difficult to understand. Choosing a very fine representation, eventually, a state will be found that divides in multiple trajectories resulting in a state transition not described in the automaton [18].

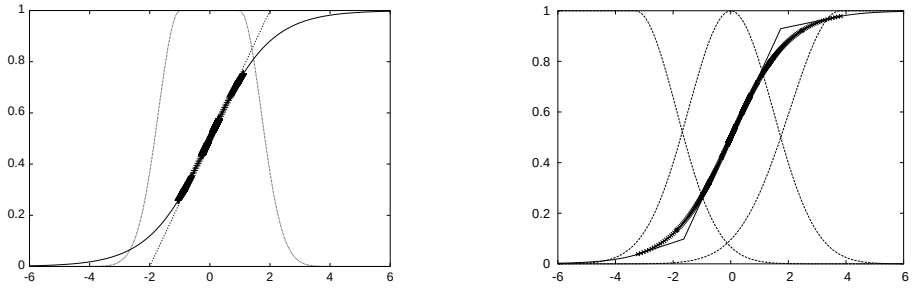


Figure 1. Activations for two neurons in the hidden layer during the simulations. The work regions of the activation function of the nonlinear neurons is shown with crosses. Neurons that work only in the central linear region of the activation function occupy only one membership function (left). Neurons 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 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 Fig. 1).

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 Fig. 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.

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, for example, $\mu_{11}\mu_{22}$. 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.

The states 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

In this paper the experiments were carried out using a set of 80 gene expression vectors for 2467 yeast genes that were selected by Eisen *et al* [8] and used with succes by Brown *et al* [7]. The data were generated from spotted arrays using samples collected at various time

points during the diaxic shift, the mitotic cell division cycle, sporulation, and temperature and reducing shocks. This data is part of the *Stanford Microarray Database* (SMD) [15] 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 and its high level of noise makes this data inadequate for training. It was necessary to extract the most important information from the original data, and only this was used as training data for the RNN.

4.1. Pre-processing

Front the great amount data to process, it was necessary to find a reduced data set that could express the main characteristics of the complete data base. 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 [21].

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

The features can be determined by the analysis of the SOM trained with the hole data. The SOM has the capacity of separate the different features present in the set. After the training we counted the number of represented by each neuron and a threshold is chose. This threshold will determine the minimum amount of data that each neuron must represents to be considered a feature. In this experiment this threshold was 35 gene expression. Using this threshold give obtained 11 features(see Fig.1). 2.

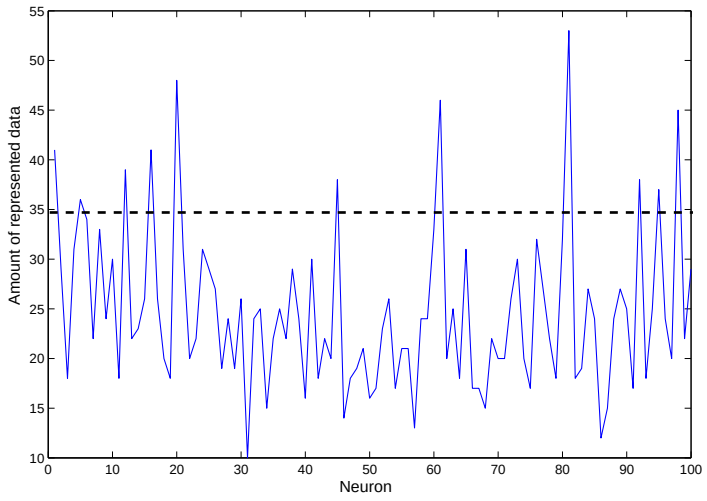


Figure 2. Amount of data represented for each neuron of SOM.

Once those features were determined and identified, each RNN pattern was formed by 36 gene expressions, the most similar data patterns to the 11 determined 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

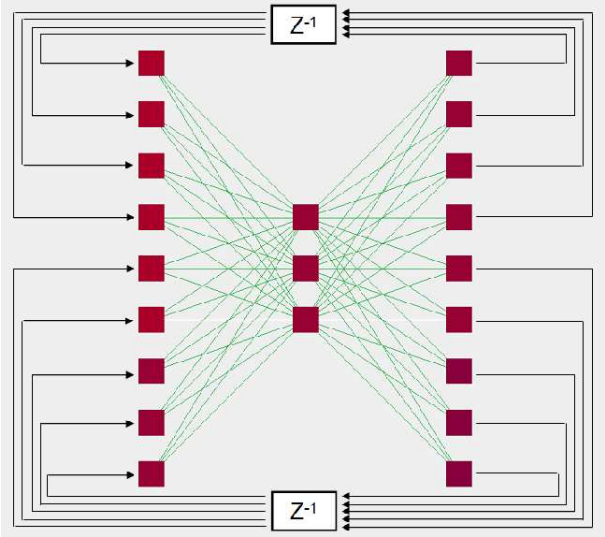


Figure 3. RNN used to the Markov Chain extraction, based on the Jordan Network's Architecture.

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. Comparative of RNNs errors.

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.

4.3. Markov Chain Extraction

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

The definition of the number of states was performed with the construction of a histogram, shown in Fig. 3, for which all membership functions were computed using all the training data. After the identification of the function with the highest degree of membership for each data, the histogram shows the 5 first states representing 95% of the sampled data.

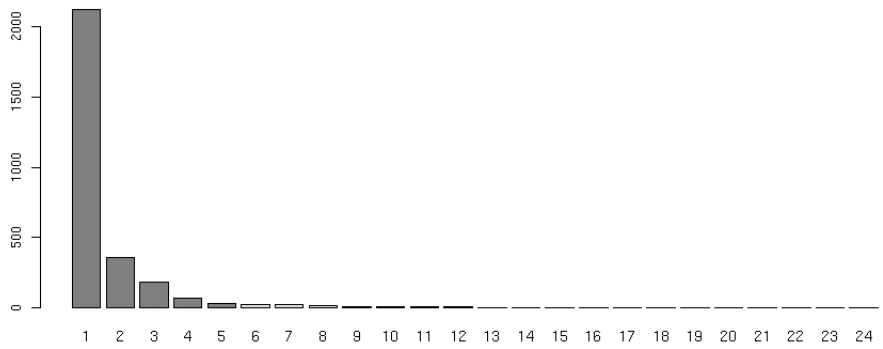


Figure 4. Histogram constructed to identify the number of Markov Chain states.

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. 5).

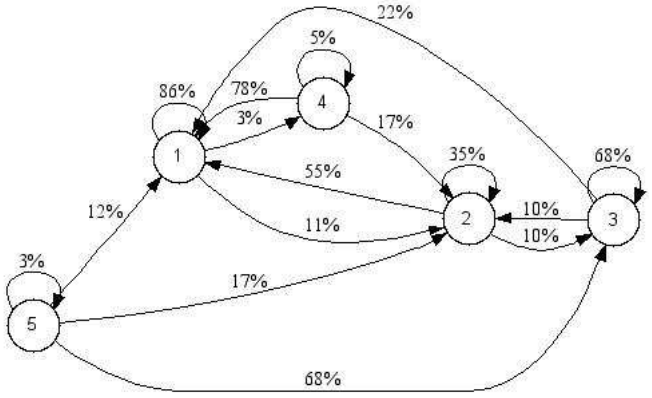
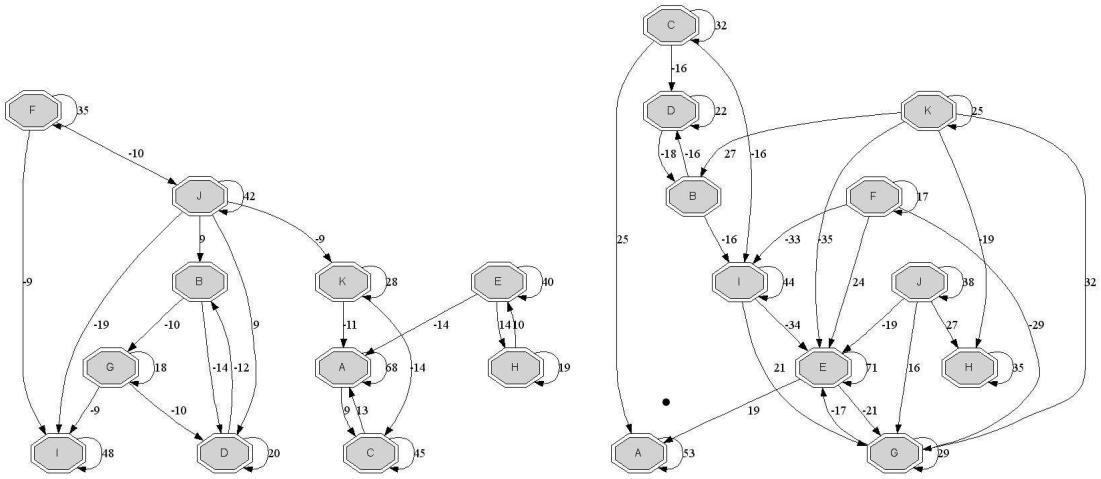


Figure 5. Markov Chain extracted from RNN with 5 hidden neurons, trained with microarray data.

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. These states represent possible states in time. That is, a state is a set of similar gene expression levels. These states are found by a special clustering procedure. Since the data is a time series, the actual gene expression levels jump from state to state. This occurs slowly and the gene expression levels may stay in a certain state for a long time. In this case, is possible to say that this is the most important state (in this case, it is the state 1).

Beyond this notion, each state is defined also by a set of equations, linear equations, that describe the interaction among genes during the state activation. Then it is possible to determine if a gene influences positively or negatively other genes, and dur-



(a) Principal State (state 1 in Fig. 5). (b) Second Principal State (state 2 in Fig. 5).

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

ing which period of time this is valid. The Fig. 6 represents the two main states. 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 this gene groups in a certain period of time of the temporal series.

5. Conclusions

The analysis of the genic expression levels with the microarray technique is becoming more common and 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 networks form. Thus, this work presented a original methodology for knowledge extraction from microarray data. We have shown the necessary data manipulation, training of the Recurrent Neural Network, extraction of information in the form of a Markov Chain and extraction of influences among gene groups.

Markov Chain and an interaction graph. The Markov Chain represents the temporal relations among genes and express the transition probabilities gene ativation levels. The interaction graph, one for each Markov State, among represents the influence of the one gene or another.

After the performed analysis, the extracted knowledge was represented in the form of a Markov Chain. This represents represent the relation among the input data, that is, among genes, identified along the time series. Each state presents a set of influence relations among genes. These relations were 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

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

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

[3] I. Kohane, A. Kho, and A. Butte. *Microarrays for an integrative genomics*. MIT Press, 2003.

[4] D. R. Pechmann and A. L. Cechin. 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, 2005.

[5] D. R. Pechmann and A. L. Cechin. 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, 2004.

[6] A. Butte. The use and analysis of microarray data. *Nature*, 2002.

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

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

[9] V. Filkov, S. Skiena, and J. Zhi. Analysis techniques for microarray time-series data. In *RECOMB*, Montreal, Canada, 2001.

[10] D. Slonim. From patterns to pathways: gene expression data analysis comes of age. *Nature Genetics Supplement*, 32, 2002.

[11] Y. Liang and A. Kelemen. Mining heterogeneous gene expression data with time lagged recurrent neural networks. *SIGKDD'02*, 2002.

[12] J. Freeman and D. Skapura. *Neural Networks. Algotirhms, Applications and Programming Techniques*. Addison-Wesley, 1992.

[13] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 282:531–537, 1999.

[14] P. Tamayo. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *PNAS*, 96:2907–2912, 1999.

[15] C. A. Ball *et al*. The stanford microarray database accommodates additional microarray platforms and data formats. *Nucleic Acids Research*, 33:D580–D582, 2005.

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

[17] C. D. Manning and H. Schutze. *Foundations of Statistical Natural Language Proc.* MIT Press, 2000.

[18] I. Cloete and J. M. Zurada. *Knowledge-Based Neurocomputing*. MIT Press, 2000.

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

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

[21] Teuvo Kohonen. *Self-organizing maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.