Design of a Self-Organizing Learning Array System

Janusz Starzyk *Tsun-Ho Liu*
School of Electrical Engineering and Computer Science
Ohio University
Athens, Ohio 45701, U. S. A

Abstract:

This paper discusses a concept of Self-Organizing Learning Array developed for programmable hardware realization. This system is designed for solving an unspecified machine-learning problems such as classification and recognition. Basic design of the array including neurons interconnections and organization is described. Symbolic values assignment method and self-organizing principle are also discussed in this paper.

1. Introduction

Evolvable Hardware is a promising approach toward autonomous and on-line reconfigurable machines [1]. They use genetic algorithms to automatically find the best hardware configuration. An implementation of genetically based reconfigurable neural network was presented in [2]. In data processing, Self-Organizing maps based on Kohonen learning algorithm are used for clustering of high-dimensional data [3]. This paper discusses an alternative method for automatic online reconfigurable machines with self-organizing principle.

Recently, a reconfigurable Self-Organizing Learning Array named SOLAR was proposed [4]. SOLAR is composed of locally pre-wired processing units called neurons. Each neuron learns from outputs of other connected neurons or directly from the external inputs.

During training, SOLAR organizes its structure according to the input training data. Each neuron is controlled by a threshold-control-input (TCI), which decides if this neuron participates in learning for the incoming data at a given time. Neurons learn in parallel by calculating and obtaining the optimum information index. Information from each neuron, such as selected connection and neuron's function, is collected to form the final structure to classify test data. Final voting is done by collecting different probabilities of a particular test data belonging to different classes and calculating the final result with a weight function.

This paper is organized as follows: Section 2 describes the structure of SOLAR network. Section 3 presents symbolic values problem and its solution. This self-organization principle of SOLAR is discussed in Section 4. A real world classification example is presented in Section 5. Finally, a conclusion is given in Section 6.

2. System Structure of SOLAR

SOLAR is an electronic system modeled on the biological brain structure. The basic element of the SOLAR system is called neuron, which is a small processing unit. This system has a feed forward organization in which outputs of earlier generated neurons feed the inputs of later generated ones. This avoids the unexpected input signal increase, which may cause system instability. The inputs of first generated neurons are connected to the primary data inputs of the learning array. Also, since biological neurons tend to interact with neighboring neurons [5], all initial interconnections between neurons in SOLAR are pseudo randomly pre-wired to near neurons with higher probability based on the Mahalanobis distance.

There are two kinds of inputs for each neuron: threshold-control-input (TCI) and data input. During training, each neuron has the capability to select input data source from the initially wired connections and perform different transformation functions (liner and nonlinear) on selected input data, while TCI decides whether the neuron should perform calculation on the incoming data or ignore it. In this architecture, every neuron can learn concurrently. Neurons select a transformation function that can result in the maximum information index based on the entropy calculation. They also select the threshold value for the output signals. The output of one neuron becomes an input of other connected neurons.

Once a neuron finished training, data selection, transformation function and the threshold value for the output with the optimum entropy value are memorized. For every training data, the same learning steps are repeated. After the whole system finished training, SOLAR sets its self-organized structure, and it is ready for classification. Some of the neurons may be left disconnected because their input space has too little information to learn from. These neurons can be reused if needed.

3. Neurons' Input Data

Input data is presented to SOLAR with n input features. Each feature represents one dimension of the whole input space. At each clock cycle, one set of n dimensional data is buffered to the input layer. The j^{th} input data with n features can be represented by a vector: $X^j=[x^j_1, x^j_2 \ldots x^j_n]$. As a result, the whole set of input data can be described by a matrix as follows.

$$\overline{X} = \begin{bmatrix} X^1 \\ \ldots \\ X^t \end{bmatrix} = \begin{bmatrix} x_1^1 x_2^1 \ldots x_n^1 \\ \ldots \\ x_1^t x_2^t \ldots x_n^t \end{bmatrix} \qquad \textbf{(3-1)}$$

where t is the length of the whole input data set.

The problem of databases containing missing data and symbolic values is very common. The incomplete data can cause problems for neuron operations, and assigning arbitrary values to represent symbols may hamper the classification effort. Therefore, blanks and symbols must be replaced by meaningful numerical values

with some transformations. Missing data problem has been addressed in [6], and the symbolic values assignment is discussed next.

If the input matrix $\overline{X}$ contains symbolic (non-numerical) data, this data can be assigned numerical values so that they are best correlated to the existing data. This can be accomplished with minimization of the determinant of the resulting covariance matrix of $\overline{X}$.

$$\overline{X} = \left[X_r \quad \tilde{X}_s \right]_{txn} \tag{3-2}$$

where $\tilde{X}_s$ is a sub-matrix with all symbolic values
X_r is a sub-matrix with all numerical values
t is the number of samples
n is the number of features

To minimize $\det[Cov(X)]$, the symbolic values, should be assigned so that the numerical vector X_s is a linear combination of vectors X_r.

$$X_s = X_r * \alpha \quad , \quad X_s \in \tilde{X}_s \tag{3-3}$$

where α is a nonzero linear combination vector.

Since this problem may not have an exact solution, the norm of the error vector E is minimized, where

$$E = X_s - X_r * \alpha \tag{3-4}$$

X_s can be replaced by the product of a binary matrix A and a vector of its symbols H.

$$X_s = AH \tag{3-5}$$

which gives the error vector

$$E = AH - X_r \alpha \tag{3-6}$$

Since the objective is to minimize the error, values of H can be obtained by applying pseudo-inverse of A.

$$H = \text{pinv}(A) X_r \alpha \tag{3-7}$$

This is a desired solution with α=1 if X_r has only one single column. If X_r has more than one column, H can be determined by minimizing the norm of the error function, shown in equation (3-8) and setting its derivatives to zero.

$$|E|^2 = E^T E \geq 0 \tag{3-8}$$

$$\frac{\partial |E|^2}{\partial H} = A^T \left[A X_r \right] \begin{bmatrix} H \\ -\alpha \end{bmatrix} = 0 \tag{3-9}$$

$$\frac{\partial |E|^2}{\partial \alpha} = X^T \left[A X_r \right] \begin{bmatrix} H \\ -\alpha \end{bmatrix} = 0 \tag{3-10}$$

Let us define matrix B as below and partition it into symbolic and numerical parts B_s and B_r.

$$B = \begin{bmatrix} A^T \\ X_r^T \end{bmatrix} * \left[A X_r \right] = \begin{bmatrix} A^T A & A^T X_r \\ X_r^T A & X_r^T X_r \end{bmatrix} = \left[B_s B_r \right] \tag{3-11}$$

The minimum error norm is obtained by solving the following equation.

$$\left[B_s B_r \right] \begin{bmatrix} H \\ -\alpha \end{bmatrix} = 0 \tag{3-12}$$

B_r can be factorized using QR factorization and its orthogonal matrix Q will be divided according to the rank of its upper triangular matrix R.

$$\left[\begin{bmatrix} Q_1^T B_s \\ Q_2^T B_s \end{bmatrix} \vdots \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix} \right] \begin{bmatrix} H \\ -\alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3-13}$$

$$\begin{cases} Q_1^T B_s H + \left[R_1 R_2 \right] \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = 0 & (3\text{-}14) \\ Q_2^T B_s H = 0 & (3\text{-}15) \end{cases}$$

Value of H can be solved by using equation (3-15) since it does not depend on α. However, H is always zero if $Q^T_2 B_s$ is a full rank matrix. A single variable in H has to be set, such as H_1=1.

$$\left(Q_2^T B_s \right) \begin{bmatrix} H_1 \\ H_{\tilde{s}} \end{bmatrix} = \left[C_1 C_{\tilde{s}} \right] \begin{bmatrix} H_1 \\ H_{\tilde{s}} \end{bmatrix} = 0 \tag{3-16}$$

where C_1 is the first column of $Q^T_2 B_s$

Then, $H_{\tilde{s}}$ can be obtained applying pseudo-inverse of $C_{\tilde{s}}$, and H can be solved as follows.

$$H = \frac{\begin{bmatrix} 1 \\ -\ pinv\left(C_{\tilde{s}} \right) C_1 \end{bmatrix}}{norm \begin{bmatrix} 1 \\ -\ pinv\left(C_{\tilde{s}} \right) C_1 \end{bmatrix}} \tag{3-17}$$

The following example illustrates the assignment of symbolic values when the rank of numerical sub-matrix X_r is larger than one. The input matrix is given as $\overline{X}$.

$$\overline{X} = \begin{bmatrix} 1 & 2 & 4 & 3 & 4 & 8 & 9 & 8 & 9 & 10 \\ 1.5 & 3 & 5 & 3 & 6 & 9 & 7 & 9 & 10 & 9.5 \\ e & a & a & b & b & d & d & c & c & c \\ 1 & 2 & 2 & 0 & 4 & 2 & -4 & 2 & 2 & -1 \end{bmatrix}^T \tag{3-18}$$

A binary matrix A which represents symbolic values and the sub-matrix X_r which contains all numerical values are

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \quad \textbf{(3-19)}$$

and

$$X_r = \begin{bmatrix} 1 & 2 & 4 & 3 & 4 & 8 & 9 & 8 & 9 & 10 \\ 1.5 & 3 & 5 & 3 & 6 & 9 & 7 & 9 & 10 & 9.5 \\ 1 & 2 & 2 & 0 & 4 & 2 & -4 & 2 & 2 & -1 \end{bmatrix}^T \quad \textbf{(3-20)}$$

Since X_r has more than one column, H can be determined by minimizing the norm of the error function and setting its derivatives to zero as shown in equation (3-8). To minimize the error norm, the matrix B can be obtained as defined in (3-11).

$$B = \left[\begin{array}{ccccc:ccc} 2 & 0 & 0 & 0 & 0 & 6 & 8 & 4 \\ 0 & 2 & 0 & 0 & 0 & 7 & 9 & 4 \\ 0 & 0 & 3 & 0 & 0 & 27 & 28.5 & 3 \\ 0 & 0 & 0 & 2 & 0 & 17 & 16 & -2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1.5 & 1 \\ 6 & 7 & 27 & 17 & 1 & 436 & 452.5 & 33 \\ 8 & 9 & 28.5 & 16 & 1.5 & 452.5 & 482.5 & 60 \\ 4 & 4 & 3 & -2 & 1 & 33 & 60 & 54 \end{array}\right] \quad \textbf{(3-21)}$$

B_s | B_r

Then, Q^T_2 is obtained using QR factorization of B_r and $Q^T_2 B_s$ is equal to

$$Q_2^T B_s = \left[\begin{array}{cc:ccc} 1.6097 & -0.2294 & -0.4622 & -0.0021 & -0.0822 \\ -0.0461 & 0.0809 & -1.0347 & 1.1301 & 0.0171 \\ -0.0669 & -0.1037 & -0.0785 & 0.0263 & 0.9823 \\ 0.3676 & 0.9404 & -1.1996 & 0.7159 & 0.0287 \\ 0.0460 & 0.1375 & -0.6420 & 0.5907 & 0.0638 \\ 0.6064 & -1.4576 & 0.0597 & 0.3330 & 0.0663 \end{array}\right] \quad \textbf{(3-22)}$$

C_1 | $C_{\tilde{s}}$

H, the normalized vector of symbolic values, can now be computed applying pseudo-inverse of $C_{\tilde{s}}$ (3-17).

$H = [0.2456 \quad 0.2824 \quad 0.6800 \quad 0.6246 \quad 0.0862]^T$

4. Self-Organization Principles

Once a network has been designed and its input data is prepared, it is ready to be trained. Unlike neural networks which have well defined organization of interconnection and neuron functions, SOLAR evolves its connections, neuron control, function and threshold during the learning phase. This is the reason the network is named the self-organizing array.

During learning, neuron first counts the total amount of training data n_t. Similar to any sequential machines, each neuron performs an operation on the selected inputs (or single input) at the rising edge of the system clock. The result may become the system output or an input of other neurons. If the TCI associated with a particular input data is high, the result of the operation is compared against a set threshold value. This means that this input data is within the subspace where the current neuron is learning. If TCI is zero, no comparison takes place since this particular input data is outside the range of the subspace where the neuron is learning. Counters in each neuron controlled by its TCI count three sets of numbers:

1. Amount of data that satisfies the threshold value: n_s
2. Amount of data belonging to a class that satisfies the threshold value: n_{sc}
3. Amount of data belonging to a class that does not satisfy the threshold value: n_{sic}

By doing so, threshold value divides the neuron's input space into two subspaces. The quality of learning of each neuron can be calculated statically by computing the information index.

In order to calculate information index, finding the probabilities of training data which falls into each subspace is required.

1. Probability of a class satisfying threshold: $P_{sc} = \frac{n_{sc}}{n_t}$
2. Probability of a class not satisfying threshold: $P_{sic} = \frac{n_{sic}}{n_t}$
3. Subspace probability (pass threshold): $P_s = \frac{n_s}{n_t}$
4. Complementary subspace probability (do not pass threshold): $P_{si} = 1 - P_s$
5. Class probability: $P_c = \frac{n_c}{n_t}$

With these calculated probabilities, information index can be obtained from equation (2-33):

$$I = 1 - \frac{\left[\sum_{sc} P_{sc}\log(P_{sc}) - P_s\log(P_s)\right] + \left[\sum_{sic} P_{sic}\log(P_{sic}) - P_{si}\log(P_{si})\right]}{\sum_c P_c \log(P_c)} \quad \textbf{(3-23)}$$

Different combination of inputs, transformation operations and TCIs can result in different information index values. Neurons perform information index calculation for different combinations, and the maximized result is obtained in order to provide an optimum separation of the input training data. When the index value becomes "1", it indicates that the neuron has solved its problem completely although it does not mean any test data can be classified correctly all the time.

Threshold value is also selected at where the maximum of information index value is located. Transformation function values are compared against threshold to separate the input space.

Figure 1 shows how the subtraction with a threshold set to -91 separates two classes.

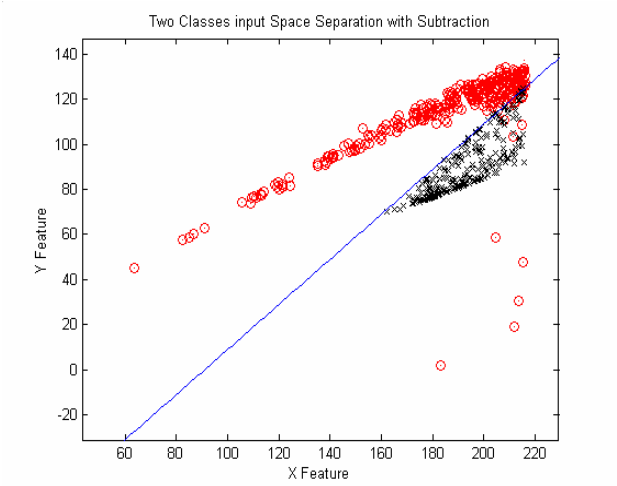

Figure 1. Input space Separation Using Subtraction Function

5. Real world classification

To illustrate the performance of SOLAR, a real world dataset, which studies if an individual's annual income exceeds $50,000, was used. This dataset is available from University of California at Irvine at "ftp://ftp.ics.uci.edu/pub/machine-learning-databases/adult/". Training data and testing data are given and already separated from the whole dataset. There is a total 45,225 cases and they are divided into two classes. All missing data and symbolic values in the dataset are recovered and replaced with numerical values using methods that have been discussed before. Since initial interconnections in SOLAR are pseudo randomly generated based on Mahalanobis distance, repeating the training and testing procedure 9 times and averaging the result gives a better statistical representation of SOLAR performance. Figure 2 illustrates one of the self-organized SOLAR structures after learning.

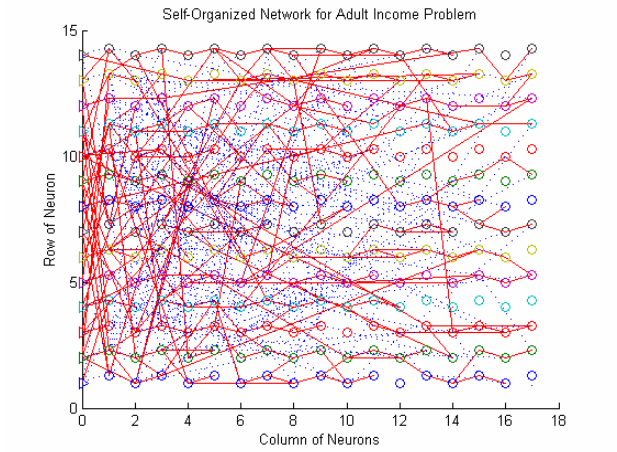

Figure 2. Self-Organized Network Structure for Adult Income Problem

Several traditional classification algorithms have been analyzed with this income dataset and their rates of miss classification were reported in the literature and are listed in Table 1 together with result for SOLAR. Although SOLAR does not have the best performance, it performs relatively well for a network that was not designed for any particular kind of classification problem.

Table 1. Miss Rate of Adult Income Classification

Algorithm	Error Rate
FSS Naïve Bayes	0.1405
NBTree	0.1410
C4.5-auto	0.1446
IDTM (Decision table)	0.1446
HOODG / SOLAR	0.1482
C4.5 rules	0.1494
OC1	0.1504
C4.5	0.1554
Voted ID3 (0.6)	0.1564
CN2	0.1600
Naïve-Bayes	0.1612
Voted ID3 (0.8)	0.1647
T2	0.1687
1R	0.1954
Nearest-neighbor (3)	0.2035
Nearest-neighbor (1)	0.2142
Pebls	Crashed

6. Conclusion

This paper discusses a concept of self-organized learning array for programmable hardware realization. SOLAR demonstrates several characteristics, which minimizes the numbers of interconnections and optimizes its performance. Neurons are locally connected which saves expensive space in VLSI design. It is self-organized and supports online training, which eliminates the tremendous off-line computation and its supportive software; each neuron in the array learns from the data concurrently and organizes its structure using information index. Testing results show its capability to solve real life problem.

Reference:

[1] T.Higuchi, T. Niwa, T. Tanaka, H. Iba, H. Garis, and T. Furuya, "Evolvable Hardware with Genetic Learning," Proc. Simulation of Adaptive Behavior, 1992.

[2] M. Murakawa, S. Yoshizawa, I. Kajitani, X. Yao, N. Kajihara, M. Iwata, T. Higuchi, "The GRD Chip: Genetic Reconfiguration of DSPs for Neural Network Processing," IEEE Trans. on Computers, vol. 48, no. 6, 1999.

[3] T. Kohonen, "The Self-Organizing Map," Neurocomput., vol. 21, pp. 1-6, 1998.

[4] J. A. Starzyk and Y. Guo, "Entropy-Based Self-Organized ANN Design Using Virtex FPGA," The Int. Conf. on Engineering of Reconfigurable Systems and Algorithms, Las Vegas, NV, June, 2001

[5] J. E. Dowling, "Creating Mind: How the Brain Works," W.W.Norton & Company, Inc., New York, 1998.

[6] J. A. Starzyk and Z. Zhu, "Software Simulation of a Self-Organizing Learning Array System," Int. Conf. on Artificial Intelligence and Soft Computing, July, 2002.