Analog VLSI Circuits for Learning Rate Adaptation in Self-Organizing Neural Networks

L. Sellami[1,2], R. W. Newcomb[2], J. M. Ferrandez[3], V. Rodellar[3], P. Gomez[3], and L. Roa[4]

1: Electrical Engineering Department, U.S. Naval Academy 105 Maryland Ave, Annapolis, MD 21402, USA. e-mail: sellami@eng.umd.edu, URL: http://web.usna.navy.mil/~sellami

2: Electrical Engineering Department, Microsystems Lab, University of Maryland, College Park, MD 20742, USA. e-mail: newcomb@eng.umd.edu, URL: http://www.ee.umd.edu/newcomb/mslab.html

3: Facultad de Informatica, Universidad Politecnica de Madrid, 28660-Madrid, Spain

4: Grupo de Biomedica, Universidad de Sevilla, Sevilla, Spain

Abstract

In this paper, we present analog VLSI circuits for the learning rate adaptation in self-organizing neural networks using the Mulier-Cherkassky learning rate adapted to the continuous-time case. The circuit design uses the solution of the Riccati equation as a basis for implementing the learning rate schedule.

1. Introduction

The self-organizing neural network (also known as the Kohonen feature map) algorithm was first introduced by Kohonen [1] and since then it has been successfully used in solving a number of pattern recognition and engineering applications. These applications include sensory mapping, robot control, vector quantization, speech and pattern recognition [2, 3, 4, 5], as well as combinatorial optimization [6] and nonparametric regression [7]. The algorithm has advantages over classical pattern recognition techniques because it utilizes the parallel architecture of a neural network and provides a graphical organization of pattern relationships. It is aimed at generating mappings from higher to lower dimensional spaces so that the relationships between the inputs are reflected in the output by establishing topological relationships among the output neurons. The map preserves the topology of the input space in the sense that nearby neurons respond to nearby stimuli.

The major drawback of the self-organizing algorithm is that it does not provide specific forms for the learning rate and the neighborhood function; these are chosen empirically to suit a particular application. It has been documented in the literature [8] that as a result of this choice several flaws can occur during training which lead to a non-optimal network. In some instances, the final organization of the feature map can depend on the learning rate and the order in which training patterns are presented to the network. In other instances, the organization of the map is determined mostly by the most recent patterns, thus causing an under-utilization of the data. Recently, a new statistical method for obtaining mathematical expressions for the learning rate has been reported by Mulier and Cherkassky [8] which eliminates some of the flaws associated with the original self-organizing algorithm. Consequently, it becomes possible to design better self-organizing networks in VLSI form to meet real-time applications.

In this paper, we present analog VLSI circuits for learning rate adaptation in self-organizing neural networks using Mulier's and Cherkassky's learning rate adapted to the continuous-time case. The circuit design uses the solution of the Riccati equation as a basis for implementing the learning rate schedule. The motivation for this work is the design of biologically inspired speech recognition systems [9]. These self-organizing neural networks are very valuable because their behavior resembles that of real neurons and follows their learning dynamics. In addition, these networks can be easily adapted for a new speaker by presenting new words to the network and letting the network relax to new equilibrium points via linear vector quantization fine tuning [9].

The paper is organized into five sections. In section 2, the Kohonen self-organizing algorithm is reviewed and its non-optimality is pointed out due to the empirical choice of the learning rate and the neighborhood function. Section 3 describes the Mulier and Cherkassky statistical method for determining the learning rate schedule independently of the choice of

0-7803-4859-1/98 $10.00©1998 IEEE

the neighborhood function. This is shown to lead to an optimal self-organizing algorithm. Based on the Mulier and Cherkassky algorithmic approach, in section 4 analog VLSI circuits are proposed to emulate the learning rate adaptation. Also included are PSpice simulation results to verify the theory. This is followed by a discussion of the results and the design methodology in section 5.

2. The Kohonen Self-Organizing Neural Networks

The Kohonen self-organizing neural network is a two-layered network that can organize a topological map from a random initial point by finding relationships among the input patterns presented to it. The network is composed of an input layer and a competitive layer (which takes the form of a two-dimensional rectangular neuron array), and learns feature mapping without supervision. All the connections are from the input layer to the competitive layer, resulting in a fully connected network. Each neuron in the input layer receives the corresponding data entry of the input pattern which is then fed to the competitive layer. In this layer, the input data are weighted according to a self-organizing weight learning rule, then added to activate a (winning) neuron. In turn, the activated neurons serve to classify input patterns in the sense that relationships (similarities or differences) among input patterns are mapped into spatial relationships (close or far) among neurons in the competitive layer.

The operation of the Kohonen feature map uses the repeated application of a two-step learning algorithm to organize neurons in the competitive layer. At each iteration, first the winning or closest neuron is found. Second, after the winning neuron is identified, the neighborhood around it, composed of the neurons that are close to it, is determined and the weights of the neurons located in this neighborhood, including the winning neuron, are updated.

The identification of the winning neuron is achieved by calculating the Euclidean distance between the input vector and the weight vector for each neuron in the competitive layer. The winning neuron is the one for which the distance is minimum. Mathematically, this is expressed as [8]:

$$i(k) = \arg_j \min \left[\sum_{n=1}^{N} (x_n(k) - w_{n,j}(k))^2 \right] \qquad (1)$$

where $i(k)$ designates the winning neuron at iteration k, $x_n(k)$ the n^{th} component of the input pattern (vector) $X(k) = [x_1(k), x_2(k), \cdots, x_N(k)]^T$ presented to the network at iteration k, N its dimension, and $w_{n,j}(k)$ the weight coefficient of the connection between neuron j in the competitive layer and neuron n in the input layer also at iteration k. The weight coefficients affecting the winning neuron and its neighborhood undergo adaptation while other weights remain the same. Random initial weight coefficients are used for the training and adaptation is performed according to the following learning rule [4, 8]:

$$\begin{aligned} W_j(k) &= W_j(k-1) + \\ &\quad \beta(k) C\left(i(k), j, k\right) [X(k) - W_j(k-1)] \end{aligned} \qquad (2)$$

Here $W_j(k)$ is the vector of weights associated with neuron j in the competitive layer, evaluated at iteration k, β the learning rate, and C the neighborhood function. The radius of the neighborhood is typically large at the beginning and shrinks as learning progresses. The learning rate is a decreasing function of k and an example of a neighborhood function yielding good practical results in a series of simulations [3] is a Gaussian function of j with mean $i(k)$ and standard deviation decreasing with k. These functions are given by

$$\beta(k) = \beta_{initial} \left[\frac{\beta_{final}}{\beta_{initial}} \right]^{\frac{k}{k_{max}}} \qquad (3)$$

$$C(i,j,k) = \exp\left(\frac{\|i-j\|^2}{(\beta(k) S_0)^2} \right) \qquad (4)$$

where $0 < \beta_{final} < \beta_{initial} < 1$, k_{max} the total number of iterations, and S_0 the number of neurons per dimension in the competitive layer.

By letting

$$a_j(i(k), k) = \beta(k) C(i(k), j, k) \text{ with } 0 \le a_j \le 1 \qquad (5)$$

the weight update rule of (2) becomes

$$W_j(k) = [1 - a_j(i(k), k)]\, W_j(k-1) + a_j(i(k), k) X(k) \qquad (6)$$

The above equation is recursive but can be made non-recursive by expressing $W_j(k)$ in terms of the initial condition $W_j(0)$ and the input data points. The resulting equation is [8]

$$\begin{aligned} W_j(k) &= \prod_{r=1}^{k} [1 - a_j(i(r), r)] W_j(0) \\ &+ \prod_{r=2}^{k} [1 - a_j(i(r), r)] a_j(i(1), 1) X(1) \\ &+ \vdots \end{aligned}$$

$$+ \quad \prod_{r=n+1}^{k} [1 - a_j(i(n), n)] a_j(i(n), n) X(n)$$

$$\vdots$$

$$+ \quad a_j(i(k), k) X(k) \tag{7}$$

which can be written in a compact form as follows

$$W_j(k) = d_j(k) W_j(0) + \sum_{n=1}^{k} d_j(k, n) X(n) \tag{8}$$

Comparing (8) and (9), it is seen that the coefficient $d_j(k)$ represents the contribution of the initial position of neuron j on the position of this same neuron at time k. Also, the $d_j(k, n)$ coefficients represent the contribution of the data patterns presented at time n to neuron j. It is this noniterative form of the weight update rule (equation (8)) that Mulier and Cherkassky use as a basis for their statistical analysis.

3. The Mulier-Cherkassky Learning Rate

Although the self-organizing algorithm presents several advantages compared to other algorithms for a number of applications, it does not provide a mathematical way of choosing the learning rate and the neighborhood function. These are set up empirically to meet the needs of the application at hand Moreover, recent statistical analyses of the self-organizing algorithm reveal several characteristic flaws associated with this ad hoc choice of learning rate and neighborhood function [8]. In their analysis and experimentation with the algorithm (using learning rate and neighborhood functions of (3) and (4)), Mulier and Cherkassky discovered that the final organization of neurons in the competitive layer (neuron location) was influenced by the order of presentation of the training patterns, the learning rate, and neighborhood function. They reported that a bad choice of the learning rate, for instance, could result in a feature map where the location of a neuron is determined mostly by the most recent patterns presented to the network. This under-utilization of the data is a serious problem when using large training data sets. Another problem which occurs with the inappropriate choice of the learning rate is that the same network trained with the same data would produce a different map if the learning rate is changed. These findings show that the choice of the learning rate is very critical for obtaining an optimum feature map.

To remedy some of these problems, Mulier and Cherkassky suggested a new method for determining an optimal learning rate schedule that allows equal contributions from all the training patterns to the final topology of the feature map. They reported that the results of self-organization obtained with their method show the position of almost 100% of the neurons is independent of the order of presentation of the training samples versus 80% in the empirical case. The method is based on the assumption that the data set is finite and generated by a stationary process. Also, the derivation of the learning rate is made independent of the neighborhood function and the initial condition (initial neuron locations) through a normalization scheme [8]. These conditions are expressed as follows:

$$\sum_{j=1}^{J} d_j(k_{max}, n) = \frac{J}{k_{max}} \text{ for } n = 1, \cdots, k_{max} \tag{9}$$

$$\sum_{j=1}^{J} C(i, j, n) = 1 \text{ for } n = 1, \cdots, k_{max} \text{ and } i = 1, \cdots, J \tag{10}$$

$$\sum_{j=1}^{J} d_j(k_{max}) \approx 0 \tag{11}$$

where J is the total number of neurons in the feature map. The optimum learning rate, i.e., one that satisfies constraints (9)-(11), is determined through a numerical search. This is done by solving for the learning rate $\beta(k)$ as a discrete-time function defined for $k = 1, 2, \cdots, k_{max}$ from equations (5) and (7) while incorporating the constraints of (9)-(11). The numerical results are curve-fitted to yield [8]

$$\beta(k) = \frac{1}{m(k-1)+1} \tag{12}$$

$$m = \frac{1 - \frac{J}{k_{max}}}{J - \frac{J}{k_{max}}} \tag{13}$$

Note that since $J < k_{max}$, $0 < m < 1$.

4. Analog VLSI Circuits for Learning Rate Adaptation

While much of the work has been theoretical, efficient implementations of self organizing networks are required to meet real-time applications. One of the key computations in these networks is learning rate adaptation used in updating the weights. In most VLSI implementations reported in the literature, weights are digitally programmed rather than updated using analog circuits. Here we present CMOS analog circuits for the learning rate adaptation. We point out that in

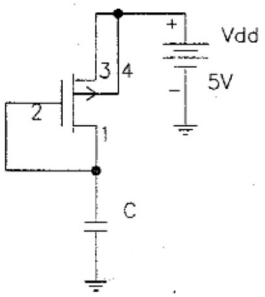

Figure 1: The basic PMOS transistor circuit.

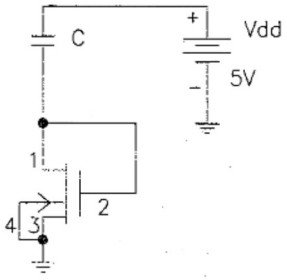

Figure 2: The basic NMOS transistor circuit.

terms of silicon area, speed, and power consumption, which are key issues in large scale real-time applications, analog implementations have unquestionable advantages over their digital counterparts.

Continuous-time Learning Rate

We take the continuous-time version of the learning rate update rule of (12) as the basis for the MOS circuit implementation

$$\beta(t) = \frac{1}{m(t-1)+1} \tag{14}$$

We note that this equation can be viewed as the solution of the Riccati equation

$$\dot{\beta} = -m\beta^2, \text{ with } \beta(0) = \beta_o \tag{15}$$

whose solution is given by

$$\beta(t) = \frac{\beta_o}{m\beta_o t + 1}, \text{ with } \beta_o = \frac{1}{1-m} \tag{16}$$

Since we desire $\beta(t) > 0$, m must satisfy $0 < m < 1$.

The Basic PMOS Circuit

Using voltage as the quantity being adapted, equation (14) can be implemented by the circuit of Figure 1 which consists of a capacitor (connected to ground) in series with a PMOS transistor. The capacitor provides the dynamics and the PMOS transistor the square law when operated in the saturation region. The current law for this circuit is

$$\begin{aligned} C\dot{v} &= -i_D \\ &= \frac{K_{P_p}}{2}\frac{W}{L}\left(V_{dd} - v + V_{TO_p}\right)^2 1(V_{dd} - v + V_{TO_p}) \end{aligned} \tag{17}$$

Here i_D is the drain current, $1(.)$ the unit step function, v the voltage across the capacitor, V_{TO_p} the threshold voltage, K_{P_p} the transconductance parameter of the transistor, and W and L the width and length of the transistor channel. Let

$$x = V_{dd} - v + V_{TO_p} \text{ and } a = \frac{K_{P_p}}{2C}\frac{W}{L} \tag{18}$$

then equation (17) becomes

$$\dot{x} = -ax^2 1(x) \tag{19}$$

The solution of (19) is

$$x = \frac{x(0)}{ax(0)t + 1}, \text{ with } x(0) = \frac{1}{1-a} \tag{20}$$

Comparing (20) to (16) and taking into account the fact that the variable t in (20) represents device time (chosen here to be in micro-seconds) while in (16) this same variable represents normal time (in seconds), we get

$$m = T_{scale} \times a = T_{scale}\frac{K_{P_p}}{2C}\frac{W}{L} \tag{21}$$

where T_{scale} $(= 10^{-6})$ is the scaling factor that allows the conversion from normal time to device time. The value of the capacitance C is a free parameter and is here set to yield reasonable values for W/L.

The Basic NMOS Circuit

Following the design methodology for the PMOS transistor circuit of the previous section, equation (16) can also be implemented with a NMOS transistor circuit as shown in Figure 2. Here it is the drain of the transistor that is grounded rather than the capacitor. Again working with voltage as the quantity being adapted and operating the transistor in the saturation region, the current law equation for the circuit is

$$C\frac{d(V_{dd}-v)}{dt} = i_D = \frac{K_{P_n}}{2}\frac{W}{L}(v - V_{T0_n})^2 1(v - V_{T0_n}) \quad (22)$$

where again $1(.)$ is the unit step function, v is the voltage between drain and source, V_{T0_n} the threshold voltage, K_{P_n} the transconductance parameter of the transistor, and W and L the width and length of the transistor channel. By letting

$$x = v - V_{T0_n} \Longrightarrow \frac{dx}{dt} = -\frac{d(V_{dd}-v)}{dt} \quad (23)$$

equation (22) is rewritten as

$$\dot{x} = -ax^2 1(x) \text{ with } a = \frac{K_{P_n}}{2C}\frac{W}{L} \quad (24)$$

Using the results obtained for the PMOS transistor circuit, the solution of (24) is given by (20), which compares with (16) for m as given by (21).

CMOS Level Shifter Circuit for Riccati Equation

Both the PMOS and NMOS transistor circuits of Figures 1 and 2 implement the learning rate update rule of (16) as a solution of the Riccati equation. But since the learning rate is represented by $x = V_{dd} - v + V_{TO_p}$ in the PMOS circuit and $x = v - V_{TO_n}$ in the NMOS circuit, which decay to 0 as $t \to \infty$ in theory, x cannot be measured directly from these circuits. This is because in the PMOS circuit, the output is the voltage across the capacitor and in the NMOS circuit, the output is the drain-source voltage. Consequently, we modify these circuits to allow for x to be measured as the output voltage. In Figure 3 we show a modified circuit with the basic PMOS circuit; a similar circuit can be constructed with the basic NMOS circuit by substituting the latter for the PMOS block. We now give the analysis for this circuit. The drain currents for M_1 (PMOS) and M_2 (NMOS) are

$$i_{D_p} = -\frac{K_{P_p}}{2}\frac{W_p}{L_p}(V_{dd} - v_{out} - |V_{TO_p}|)^2 \quad (25)$$

$$i_{D_n} = \frac{K_{P_n}}{2}\frac{W_n}{L_n}(v_{in} - V_{TO_n})^2 \quad (26)$$

Equating the two currents and solving for v_{out} yields

$$v_{out} = V_{dd} - |V_{TO_p}| + KV_{TO_n} - Kv_{in} \quad (27)$$

where

$$K = \sqrt{\frac{K_{P_n}}{K_{P_p}}\frac{W_n}{W_p}\frac{L_n}{L_p}} \quad (28)$$

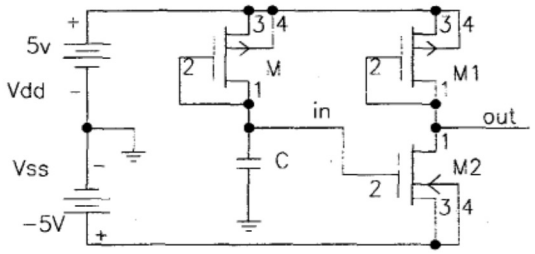

Figure 3: CMOS level shifter circuit for Riccati equation.

Since the voltage across the capacitor $v = v_{in} + V_{ss}$ and $V_{ss} = -V_{dd}$, we get

$$v_{out} = K\left[\left(\frac{V_{dd}}{K} - V_{dd}\right) - \left(\frac{|V_{TO_p}|}{K} - V_{TO_n}\right) - v\right] \quad (29)$$

Note that V_{out} of (29) does not quite match x of (18). However, with the proper choice of K and the fact $V_{TO_n} \approx |V_{TO_P}|$, V_{out} can approximate x up to a constant ($= K$).

5. Simulation Results

To verify the theory, the circuits of Figures 1, 2, and 3 are simulated with PSpice, where the transistor models from MOSIS run N21H of 04/28/93 are used. The key transistor model parameters are $K_{P_n} = 5.048 \times 10^{-5}$, $K_{P_p} = 1.908 \times 10^{-5}$, $V_{TO_n} = 0.858V$, and $V_{TO_p} = -0.889V$. In all simulations, appropriate values of capacitance C, W, and L are used to yield $0 < m < 1$. For the basic PMOS and NMOS circuits, of Figures 1 and 2, $C = 2pF$, $W = 10\mu$, and $L = 10\mu$. For the circuit of Figure 3, $C = 2pF$, $L = 10\mu$, $W = 10\mu$, $L_p = 10\mu$, $W_p = 47\mu$, $L_n = 20\mu$, and $W_n = 10\mu$. With this choice of parameters, the calculated $K = 0.53$. The simulation results are given in Figures 4, 5, and 6 where Figure 4 shows the plot of $V_{dd} - v$ decaying to $-V_{TO_p}$, Figure 5 shows the plot v decaying to V_{TO_n}, and Figure 6 shows the plot of v_{out} decaying to 0.

6. Conclusions

In this paper, we have presented analog VLSI circuits for learning rate adaptation in self-organizing neural networks. The learning rule used is the continuous-time version of that of Mulier and Cherkassky, which has been proven to lead to optimal feature maps. In our

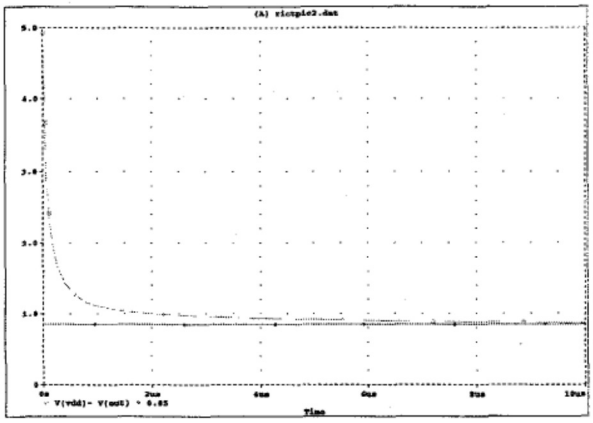

Figure 4: $V_{dd} - v$ for the basic PMOS circuit.

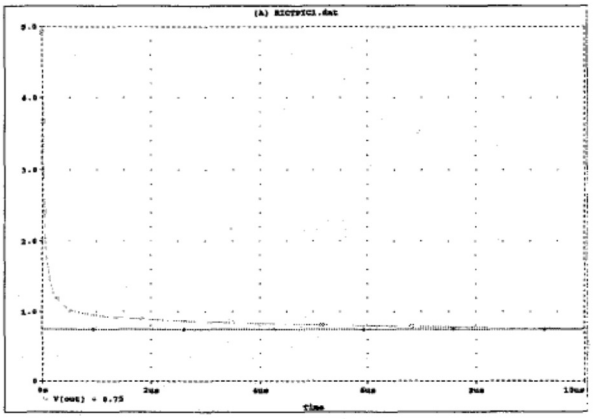

Figure 5: v for the basic NMOS circuit.

analysis, we have expressed this learning rate update rule as the solution of a Riccati equation and, consequently, we have developed MOS transistor circuits to implement this solution. PSpice simulations were run and the results obtained verify the theory for the basic PMOS and NMOS circuits. For the level shifter circuit v_{out} satisfies the Riccati equation of (19) if we redefine x in (18) as x/K and reflect this change in (17) by redefining a in (18) as aK^2.

Acknowledgments

Assistance of NATO Grants # CRG 87-0395 and # CRG 960053 in allowing interchanges of ideas between the authors is gratefully acknowledged.

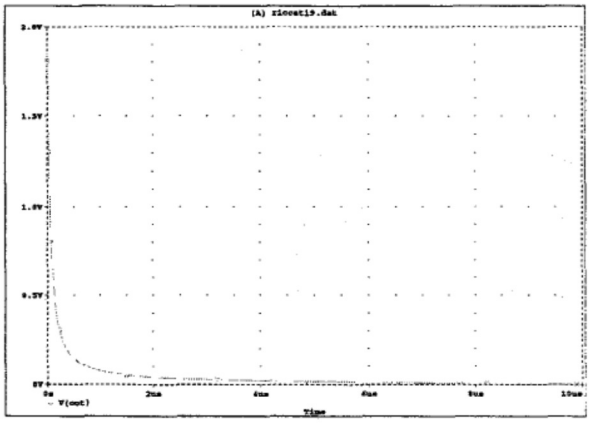

Figure 6: v_{out} for the CMOS level shifter circuit.

References

[1] T. Kohonen, *Self-Organization and Associative Memory*, 3rd edition, Springer-Verlag, 1989.

[2] T. Kohonen, "Speech Recognition Based on Topology-Preserving Neural Maps," *in Neural Computer architectures*, Igor Aleksander, Ed., North Oxford Academic, London, England, 1989.

[3] T. Kohonen, "The Self-Organizing Map," *Proceedings of IEEE*, Vol. 78, No. 9, pp. 1464–1480, 1990.

[4] T. Kohonen, *Self-Organizing Maps*, 2nd edition, Springer-Verlag, 1997.

[5] G. Carpenter and S. Grossberg, *Pattern Recognition by Self-Organizing Neural Networks*, Cambridge, Massachusetts, MIT Press, 1991.

[6] J. C. Fort, "Solving a Combinatorial Problem Via Self-Organizing Process," *Biological Cybernetics*, Vol. 59, pp. 33–40, 1988.

[7] V. Cherkassky and H. Lari-Najafi, "Constrained Topological Mappings for Nonparametric Regression Analysis," *Neural Networks*, Vol. 4, pp. 27–40, 1991.

[8] F. M Mulier and V. S. Cherkassky, "Statistical Analysis of Self-Organization," *Neural Networks*, Vol. 8, No. 5, pp. 717–727, 1995.

[9] R. M. Ferrandez, D. del Valle, V. Rodellar, and P. Gomez, "A Neural Network Hierarchical Model for Speech Recognition Based on Biological Plausibility," *Proceedings of the First International Workshop on Machine Learning, Forecasting, and Optimization*, pp. 53-64, Madrid, Spain, July 1996.