TẠP CHÍ KHOA HỌC ĐẠI HỌC HẠ LONG
Scientific Journal of Ha Long University
http://uhl.edu.vn/

LUẬT HỌC PERCEPTRON CẢI TIẾN VÀ KHẢ NĂNG ỨNG DỤNG TRONG MẠNG NƠRON TẾ BÀO

Dương Đức Anh[1*], Nguyễn Tài Tuyên[2], Nguyễn Thanh Tùng[3], Nguyễn Quang Hoan[2]

[1]Viện Nghiên cứu Điện tử, Tin học, Tự động hóa
[2]Học viện Công nghệ Bưu chính Viễn thông
[3]Học viện Công nghiệp Phần mềm và nội dung số

** Email: datdh1@gmail.com*

Ngày nhận bài: 10/11/2022 *Ngày nhận bài sửa sau phản biện: 05/12/2022* *Ngày chấp nhận đăng: 15/12/2022*

TÓM TẮT

Bài báo này cải tiến luật học Perceptron để có thể áp dụng được cho các mạng nơron truy hồi nói chung và mạng nơron tế bào (CNNs: Cellular Neural Networks) nói riêng, khi mà phiên bản gốc chỉ dùng cho các mạng nơron truyền thẳng. Để thực hiện điều này, ta ghép tín hiệu vào, tín hiệu phản hồi và độ lệch thành một tín hiệu vào tổng quát; phần còn lại luật học tiến hành như phiên bản gốc. Tuy nhiên, do đặc thù của mạng nơron tế bào, một số tham số cũng được bổ sung và cải biên ít nhiều. Một vài ví dụ cũng được tiến hành trong bài báo nhằm trực quan hóa ý tưởng.

Từ khóa: *luật học Perceptron, mạng nơron tế bào, mạng truy hồi, phương pháp thử-sai-chỉnh*

MODIFIED PERCEPTRON LEARNING RULE AND APPLICATION ABILITIES FOR CELLULAR NEURAL NETWORKS

ABSTRACT

This paper modifies the Perceptron learning rule in order to apply to all recurrent neural networks in general and cellular neural networks in particular since the original Perceptron learning rule was only used for feedforward neural networks. The idea is as follows, we link input, feedback output, and the bias of the cellular neural network to become new general input. The next step of the process can be implemented as the original Perceptron learning rule. However, cellular neural networks characterized by some features and several parameters in the learning rule that modifies the Perceptron can be added. Some examples are proposed in the paper to visualize the idea.

Keywords: *cellular neural networks, Perceptron learning rule, recurrent neural networks, Trial-Error-Correct approach*

1. ĐẶT VẤN ĐỀ

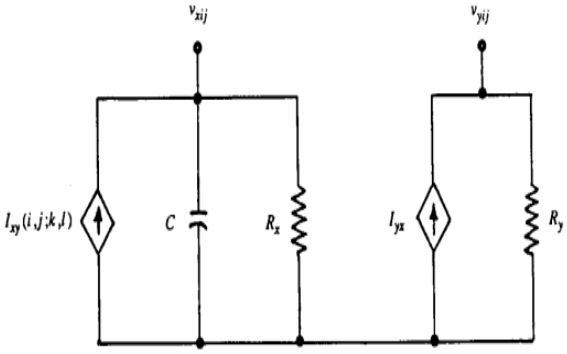
Năm 1960, Frank Rosenblatt đề xuất luật học xác định bộ trọng số cho mạng truyền thẳng một lớp (Nguyễn Quang Hoan, 2022). Đối với CNNs thuộc họ mạng phản hồi, thông thường dùng luật học Hebb để tìm trọng số phản hồi (Chua & Yang, 1988; Nguyễn Quang Hoan, Nguyễn Tài Tuyên &

Dương Đức Anh, 2020). Để tìm được tập đầy đủ bộ tham số của CNNs gồm [*A, B, I*], nhiều tác giả đã đề xuất các phương pháp khác nhau; mỗi phương pháp có những ưu, nhược điểm cũng như các phạm vi ứng dụng khác nhau đã được Güzeliş và các cộng sự phân tích và đánh giá (Zou & cs, 1990; Chua & Thiran, 1991; Güzeliş & cs, 1998; Mizutani, 1994; Sergey, 2001). Trong phạm vi bài báo này, chúng tôi thử nghiệm phương pháp gộp bộ ba tham số của CNNs thành chỉ một ma trận tương đương $\boldsymbol{W} = [\boldsymbol{A^T}\ \mathbf{B^T}\ \mathbf{I}]$. Khi đó, CNNs được xem như tương đương với một mạng truyền thẳng và hoàn toàn có thể áp dụng thuật toán Perceptron truyền thống để tính toán. Thuật toán cải tiến này áp dụng cho CNNs – một loại mạng hồi quy, nên có thể gọi là luật học Perceptron hồi quy.

2. NHẬN DẠNG BỘ THAM SỐ CNNs

2.1. Cấu trúc mạng nơron tế bào

CNNs được xây dựng và phát triển dựa trên kiến trúc của một mạch điện tương tự (Chua & Yang, 1988). Mỗi một mạch điện của CNNs được coi là một tế bào, tại đó chứa các thành phần tuyến tính như tụ điện *C*, điện trở R_x và thành phần phi tuyến bao gồm nguồn điều khiển phi tuyến và các nguồn ngoài (Hình 1).



Hình 1. Mạch tế bào đơn giản của CNNs

Hệ phương trình động học của CNNs cơ bản (Chua & Yang, 1988), như sau:

2.1.1. Phương trình trạng thái

$$C\frac{dx_{ij}(t)}{dt} = -\frac{1}{R_x}x_{ij}(t) + I + \sum_{(k,l)\in N_r(i,j)} A(i,j;k,l)y_{kl}(t) + \sum_{(k,l)\in N_r(i,j)} B(i,j;k,l)u_{kl}$$
$$0 \le i,k \le M;\ 0 \le j,l \le N \quad (1)$$

Trong đó:

r: bán kính lân cận của tế bào *C(i, j)*. Bài toán này sử dụng bán kính lân cận *r =1*, khi đó *C(i, j)* chịu tác động của 08 tế bào lân cận và sự phản hồi của chính *C(i, j)*.

C, R_x: tụ điện, điện trở của CNNs

$A(i,j;k,l), B(i,j;k,l)$: ma trận trọng số mô tả việc kết nối giữa tế bào *C(i, j)* với tế bào lân cận *C(k, l)* của tín hiệu đầu ra, đầu vào một cách tương ứng; kích thước (*3x3*).

(*i, j*), (*k, l*): thể hiện vị trí của tế bào trong CNNs có kích thước (*MxN*).

M, N: số tế bào của CNNs theo chiều ngang, và chiều dọc một cách tương ứng.

I: ma trận ngưỡng, kích thước *(1x1)*.

$x_{ij}(t) \in R^n$: trạng thái tế bào *(i, j)*, kích thước (*MxN*).

$y_{ij}(t) \in R^n$: ma trận tín hiệu đầu ra của mạng; kích thước (*MxN*).

$u_{ij} \in R^n$: tín hiệu đầu vào của mạng; kích thước (*MxN*).

2.1.2. Hàm tương tác đầu ra

$$y_{ij}(t) = \frac{1}{2}\left(|x_{ij}(t)+1| - |x_{ij}(t)-1|\right) \quad (2)$$

2.1.3. Điều kiện ràng buộc để CNNs ổn định

Ma trận phản hồi cần có tính đối xứng:

$$A(i, j; k, l)=A(k, l; i, j)$$

2.2. Luật học Perceptron cải tiến cho CNNs

Thuật toán Perceptron cải tiến RPLA dùng để tính toán các trọng số của CNNs chuẩn trong trạng thái ổn định (Slavova, 2003). Tại đây C. Güzeliş đề xuất phương án gộp các các ma trận phản hồi ***A***, ma trận đầu vào ***B***, và ma trận ngưỡng ***I*** thành ma trận trọng số ***W*** (Güzeliş & Karamahmut, 1994). Khi CNNs chuẩn đạt trạng thái ổn định thì (1) có vế trái bằng 0. Khi đó, (1) có thể viết lại như sau:

$$\frac{1}{R_x}x_{ij}(t) = I + \sum_{(k,l)\in N_r(i,j)} A(i,j;k,l)y_{kl}(t) + \\ + \sum_{(k,l)\in N_r(i,j)} B(i,j;k,l)u_{kl} \tag{3}$$

Ma trận ***W***:

$$W = [\,A^T \;\; B^T \;\; I\,] \tag{4}$$

và đầu ra của CNNs có dạng:

$$y_{ij}(t) = [y_{kl}^T \;\; u_{kl}^T \;\; 1] \tag{5}$$

Dễ thấy, phương trình (3) tại trạng thái ổn định tương đương với mạng Perceptron truyền thẳng:

$$Ax_{ij}^s(\infty) = \left[Y_{i,j}^s\right]^T * w \\ = \left[\left[y_{i,j}^s\right]^T \;\; \left[u_{i,j}^s\right]^T \;\; 1\right]^T * w \tag{6}$$

Khi véc-tơ trạng thái ***X*** của CNNs đạt ổn định, có thể chọn một bộ tham số mẫu bao gồm đầu vào $u_{i,j}^s$ và đầu ra $y_{i,j}^s(\infty)$ làm các tham số khởi tạo, trong đó *s* là số mẫu huấn luyện mạng:

$$y_{i,j}^s(\infty) = \begin{bmatrix} y_{i-1,j-1}^s(\infty) + y_{i+1,j+1}^s(\infty) \\ y_{i-1,j}^s(\infty) + y_{i+1,j}^s(\infty) \\ y_{i-1,j+1}^s(\infty) + y_{i+1,j-1}^s(\infty) \\ y_{i,j-1}^s(\infty) + y_{i,j+1}^s(\infty) \\ y_{i,j}^s(\infty) \end{bmatrix} \tag{7}$$

$$u_{i,j}^s = \begin{bmatrix} u_{i-1,j-1}^s + u_{i+1,j+1}^s \\ u_{i-1,j}^s + u_{i+1,j}^s \\ u_{i-1,j+1}^s + u_{i+1,j-1}^s \\ u_{i,j-1}^s + u_{i,j+1}^s \\ u_{i,j}^s \end{bmatrix} \tag{8}$$

Với bộ tham số trên, giả sử CNNs có đầu ra lưỡng cực (âm (-1), dương (+1)), tức là kết quả từ hàm tương tác đầu ra, hàm dấu *sgn(.)*, khi cho tín hiệu đầu vào qua bộ trọng số và hàm tương tác đầu ra thì nhận được đầu ra của CNNs ổn định như sau:

$$y_{i,j}^s(\infty) = sgn[[Y_{i,j}^s]^T * w] \\ = sgn[[y_{ij}^s]^T * a + [u_{ij}^s]^T * b + I] \tag{9}$$

Việc tính toán trọng số mạng Perceptron này được thực hiện dựa trên phương pháp Thử–Sai–Chỉnh từ bộ dữ liệu mẫu cho trước. Các sai lệch được tính theo phương pháp bình phương tối thiểu (Güzeliş & cs, 1998):

$$\varepsilon\left[w[n]\right] = \frac{1}{2}\sum_{i,j,s} y_{i,j}^s(\infty) \times (y_{i,j}^s(\infty) - d_{i,j}^s) \\ = \sum_{i,j\in D^+} y_{i,j}^s(\infty) - \sum_{i,j\in D^-} y_{i,j}^s(\infty) \tag{10}$$

trong đó:

$\varepsilon\left[w[n]\right]$: tổng véc-tơ sai số đầu ra rút gọn;

$d_{i,j}^s$: giá trị đầu ra mong muốn CNNs;

$y_{i,j}^s(\infty)$: giá trị đầu ra ổn định CNNs;

D^+, D^-: miền giá trị đầu ra sai lệch tính toán dựa theo nguyên tắc sau:

$$D^+ = \begin{cases} y_{i,j}^s(\infty) = 1 \\ d_{i,j}^s = -1 \end{cases} \quad D^- = \begin{cases} y_{i,j}^s(\infty) = -1 \\ d_{i,j}^s = +1 \end{cases}$$

Khi sai số $\varepsilon[w] \geq \delta$ cho trước, thuật toán Perceptron cải tiến sẽ tính giá trị cập nhập trọng số CNNs ổn định khi chuyển sang rời rạc hóa:

$$W(n+1) = W(n) - \Delta W(n)$$
$$\Delta W(n) = \alpha \left(\sum_{i,j \in D^+} Y_{i,j}^s(n) - \sum_{i,j \in D^-} Y_{i,j}^s(n) \right) \quad (11)$$

Trong đó:

δ: giá trị sai lệch cho phép của mạng;

ΔW: sai lệch trọng số;

α: tốc độ học của mạng;

n: số bước tính toán.

Luật cập nhật trọng số CNNs ổn định:

$$w\ n+1 = w\ n\ - \alpha * \varepsilon\ W \quad (12)$$

Trình tự thực hiện các bước của luật học Perceptron cải tiến như sau:

Input: Cho bộ mẫu huấn luyện CNNs bao gồm $(u_{ij}^s, x_{ij}^s(0), d_{ij}^s)$, tốc độ học α, chu kỳ tính toán trạng thái CNNs step time. Chọn ma trận trọng số phản hồi khởi tạo của mạng đảm bảo theo tính phân cực đối xứng.

$$A\ 0 = \begin{bmatrix} a_1\ 0 & a_2\ 0 & a_3\ 0 \\ a_4\ 0 & a_5\ 0 & a_4\ 0 \\ a_3\ 0 & a_2\ 0 & a_1\ 0 \end{bmatrix};$$

Chọn ma trận đầu vào có tính chất tương tự ma trận phản hồi, và ma trận I như sau:

$$B\ 0 = \begin{bmatrix} b_1\ 0 & b_2\ 0 & b_3\ 0 \\ b_4\ 0 & b_5\ 0 & b_4\ 0 \\ b_3\ 0 & b_2\ 0 & b_1\ 0 \end{bmatrix};$$

$$I = I\ 0$$

Output: Tìm bộ tham số của CNNs

Bước 1: Chọn bộ trọng số khởi tạo của mạng CNNs

Bước 2: Tính sai lệch CNNs

Bước 2.1: Tính tổng sai lệch $\varepsilon\left[w\left[n\right]\right]$ giữa đầu ra $y_{ij}^s(\infty)$và mẫu d_{ij}^s mong muốn đã biết

Bước 2.2: Cập nhập bộ trọng số của CNNs;

Bước 2.3: Tính toán giá trị trạng thái $x_{ij}^s[n]$.

Bước 3: Kiểm tra kết quả tính toán

Bước 3.1: Tính đầu ra của mạng $y_{ij}^s[n]$;

Bước 3.2: Tính tổng sai lệch $\varepsilon[w]$ của $y_{ij}^s[n]$ và d_{ij}^s.

Bước 4: Nếu sai lệch khác không, quay lại bước 2.2 để thực hiện các bước tiếp theo.

Còn nếu sai lệch bằng không, chọn bộ trọng số đã tính toán.

Bước 5. Kết thúc.

3. THỬ NGHIỆM THUẬT TOÁN TÌM BỘ THAM SỐ CNNs

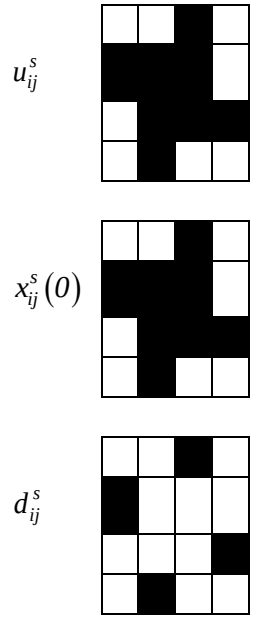
3.1. Thử nghiệm thuật toán

Input: Cho bộ mẫu của CNNs sau:

u_{ij}^s : ma trận đầu vào (*4x4*)

$x_{ij}^s(0)$ *:* ma trận trạng thái ban đầu (*4x4*)

d_{ij}^s *:* ma trận đầu ra mong muốn (*4x4*)



Hình 2. Tập mẫu cho luật học

Giả thuyết bộ mẫu phân cực như hình trên, trong đó ô màu trắng có giá trị là -1; ô màu đen có giá trị +1 đối với cả $u_{ij}^s, x_{ij}^s(0), d_{ij}^s$, tốc độ học α = 0.1.

Output: tìm bộ tham số.

Thuật toán

Bước 1: Chọn bộ trọng số khởi tạo cho CNNs cần huấn luyện.

$$A=\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix}\quad B=\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix}\quad I=0$$

Bước 2: Tính toán sai lệch CNNs

Bước 2.1: Tính sai lệch đầu ra tại bước 1.

Khi CNNs ở trạng thái ổn định, tại thời điểm ban đầu: $y_{kl}(0)=u_{kl}=x_{kl}(0)$

Tổng sai số của mạng khi đó như sau:

$$\varepsilon[w[0]]=\sum_{i,j\in D^+}y_{i,j}^s(0)-\sum_{i,j\in D^-}y_{i,j}^s(0)$$

Thay vào giá trị thực tế ta được:

$$\sum_{i,j\in D^+}y_{i,j}^s(0)=4;\ \sum_{i,j\in D^-}y_{i,j}^s(0)=0$$

Từ phương trình (10):

$$\varepsilon(w[0])=\sum_{i,j\in D^+}y_{i,j}^s(0)-\sum_{i,j\in D^-}y_{i,j}^s(0)=4$$

Bước 2.2: Cập nhập trọng số mới của mạng được tính theo (12):

$$A[1]=\begin{bmatrix}0&-0.0154&0\\-0.0154&1.9846&-0.0154\\0&-0.0154&0\end{bmatrix}$$

$$B[1]=\begin{bmatrix}0&-0.0173&0\\-0.0173&-0.0154&-0.0173\\0&-0.0173&0\end{bmatrix};\quad I[1]=-0.0018$$

Bước 2.3: Tín hiệu trạng thái $x_{ij}^s(\infty)[1]$ của mạng theo bộ trọng số tính toán mới:

$$x_{ij}\ 1=\begin{bmatrix}-1.1971&-1.2004&1.200&-1.1971\\1.200&1.1902&1.1902&-1.2004\\-1.2004&1.1902&1.1902&1.200\\-1.1971&1.200&-1.2004&-1.1971\end{bmatrix}$$

Bước 2.4: Tính toán sai lệch của giá trị đầu ra $y_{ij}^s(\infty)[1]$ với tín hiệu đầu ra mong muốn, kiểm tra điều kiện dừng thuật toán.

$$y_{ij}[1]=\begin{bmatrix}-1&-1&1&-1\\1&1&1&-1\\-1&1&1&1\\-1,&1&-1&-1\end{bmatrix}$$

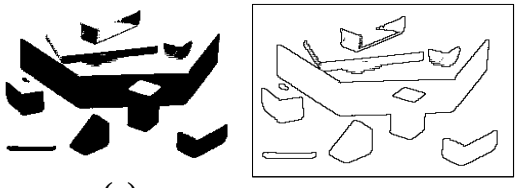
Thực tế với bộ mẫu huấn luyện trên, dựa theo chương trình tạo ra trên máy tính, sau 8 vòng lặp giá trị tổng sai lệch $\varepsilon[w[8]]=0$ (thỏa mãn điều kiện dừng của thuật toán), bộ trọng số **A, B, I** giữ không đổi tại vòng tính toán tiếp theo. Bộ trọng số của CNNs khi đó như sau:

$$A[8]=\begin{bmatrix}0&-1.4051&0\\-1.4049&0.5950&-1.4049\\0&-1.4051&0\end{bmatrix}$$

$$B[8]=\begin{bmatrix}0&-1.4478&0\\-1.4476&-1.4326&-1.4476\\0&-1.4478&0\end{bmatrix}\quad I[8]=-1.4190$$

3.2. Ứng dụng thuật toán Perceptron cải tiến cho xử lý ảnh

Phần này, nhóm tác giả thử nghiệm hai pha. Trong đó, pha một tính ma trận trọng số của CNNs dựa trên thuật toán đã nêu, pha hai dùng ma trận trọng số tính được thử bài toán nhận dạng biên ảnh dựa theo góc, cạnh như Hình 2.



(a) (b)

Hình 3. Ảnh mẫu và đường biên nhận dạng ứng dụng luật học Perceptron cải tiến

Ảnh có kích thước 32x32, với các vật thể hiển thị như Hình 3a. Sau pha một, sử dụng các ma trận trọng số *A, B, I* đã tính toán tại

mục *3.1* để xác định biên ảnh dùng mã nguồn mở mô phỏng mạng CNNs (Aggarwal, 2018). Từ bức ảnh gốc và mô hình mô phỏng CNNs chuẩn có trọng số tối ưu, tạo ra bức ảnh Hình 3b đã xử lý đường biên để làm nổi bật, hiển thị rõ ràng cạnh của các đối tượng trong bức ảnh gốc.

Như vậy, bộ trọng số CNNs tính toán tối ưu nêu trên theo phương pháp học Perceptron cải tiến đã đảm bảo tính hội tụ của bài toán nhận dạng đường biên, nhận dạng đỉnh trong việc xử lý ảnh. Từ ứng dụng này có thể thấy phương pháp xử lý ảnh sử dụng thuật toán RPLA có thể áp dụng cho các bài toán thực tế hiện nay.

4. KẾT LUẬN

Bài báo đã nêu thuật toán Perceptron cải tiến để làm rõ cách tính các trọng số. Ưu điểm của thuật toán này là giúp ta tính toán đầy đủ các thông số cốt để hệ thống chạy được với đủ các đầu vào và phản hồi của đầu ra. Việc tính toán các trọng số này đảm bảo cho CNNs ổn định. Thuật toán được đưa vào thử nghiệm trong xử lý ảnh, nhận dạng đỉnh, đường biên của đối tượng (Hình 3).

Từ kết quả của bài báo, nhóm thực hiện nhận thấy CNNs có tính ứng dụng thích hợp đối với các bài toán xử lý ảnh. Tuy nhiên, ma trận ảnh có kích thước lớn nên việc sử dụng CNNs bậc cao là cần thiết (Nguyễn Quang Hoan & cs., 2020; Nguyễn Tài Tuyên, 2022). Hướng phát triển tiếp theo là cải tiến thuật toán Perceptron truy hồi này cho CNNs bậc cao. Nhóm đang tiến hành chạy thử theo hướng nghiên cứu mới và có khả năng áp dụng cho các bài toán phức tạp hơn.

TÀI LIỆU THAM KHẢO

Aggarwal, A. (2018). *PyCNN: Image Processing with Cellular Neural Networks in Python*. Truy cập từ https://github.com/ankitaggarwal011/PyCNN.

Chua, L., O. & Yang., L. (1988). Cellular Neural Networks: Applications. *IEEE Transactions on Circuits and Systems, 35*, 1273-1290.

Chua, L., O. & Yang, L. (1988). Cellular Neural Networks: Theory. *IEEE Transactions on Circuits and Systems, 35*(10), 1257 - 1272.

Chua, L., O. & Thiran, P. (1991). An Analytic Method for Designing Simple Cellular Neural Networks. *IEEE Transactions on Circuits and Systems, 38*(11), 1332-1341.

Güzeliş, C. & Karamahmut, S. (1994). Recurrent Perceptron Learning Algorithm for Completely Stable Cellular Neural Networks. *Proceedings of the Third IEEE International Workshop on Cellular Neural Networks and Their Applications*, 177-182.

Güzeliş, C., Karamamut, S. & Genç, İ. (1998). A Recurrent Perceptron Learning Algorithm for Cellular Neural Networks. *ARI - An International Journal for Physical and Engineering Sciences, 51*, 296-309.

Mizutani, H. (1994). A New Learning Method for Multilayered Cellular Neural Networks. *Third IEEE International Workshop on Cellular Neural Networks and Their Applications*, 195-200.

Nguyễn Quang Hoan. (2022). Bộ nhớ liên kết dựa trên mạng nơron tế bào. *Tạp chí Khoa học Đại học Hạ Long, Số 03*, 90-94.

Nguyễn Quang Hoan, Nguyễn Tài Tuyên & Dương Đức Anh. (2020). Achitecture and Stability of the Second – Order Cellular Neural. *UTEHY Jounal of Science and Technology, 27*, 91-97.

Nguyễn Tài Tuyên. (2022). *Phát triển mạng nơron tế bào đa tương tác và khả năng ứng dụng*. Thư viện Quốc gia.

Sergey, P. (2001). Learning of Cellular Neural Networks. *Future Generation Computer Systems, 17*, 689-697.

Slavova, A. (2003). *Cellular Neural Networks: Dynamics and Modelling*. Springer Dordrecht.

Zou, F. , Schwarz, S. & Nossek, J., A. (1990). Cellular Neural Network Design Using a Learning Algorithm. *IEEE International Workshop on Cellular Neural Networks and Their Applications*, 74-81.