FINDING THE STABLE STATE ATTRACTOR OF THE DISCRETE HOPFIELD NETWORK USING AN OJA-LIKE NEURON

Amin Shoukry
Computer Science Department - Alexandria University
Alexandria, 21544 - Egypt
email : aminsh@alex.eun.eg

ABSTRACT

A quadratic objective (energy) function; that is logically derived from the corresponding objective function of the Discrete Deterministic Hopfield Neural Network (**DDHNN**) ; is formulated. This formulation adopts *a new look* to the operation of a **DDHNN** (with N neurons) : instead of considering the Hopfield weight matrix (**W**) as a projection operator that acts upon the state vector (**u**), *it adopts a dual view by considering the projection of the row vectors of* **W** *on* **u**. This alternative view has made possible the **realization** of the idea of finding *the stable state for a feedback attractor network by a pure feedforward network of lower complexity* at the expense of an increase in the time complexity of the proposed technique. The convergence and the complexity; in terms of storage space and execution time; of the proposed algorithms are discussed. The paper, also, points to a new **deterministic** version of the (unsupervised) *OJA constrained Hebbian learning rule* that is especially **useful** in calculating the eigenvector corresponding to the maximal eigenvalue of a *real and symmetric positive definite matrix.*

Keywords: Neural Networks, Discrete deterministic Hopfield model - Energy function - Maximal eigenvector - Principal component - Oja learning rule.

INTRODUCTION

*A.- About the Discrete Deterministic Hopfield Neural Network (**DDHNN**) .*

A **DDHNN** [1,2] is an auto-associative network of artificial neurons with symmetric connections and asynchronous update strategy. The network activity dynamics converge to a steady state corresponding to a stationary point that is a *minimum* of the following objective function :

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}\, u_i\, u_j - \sum_{i=1}^{N} \theta_i\, u_i \quad (1)$$

where, **W** is an N*N (symmetric with zero diagonal) weight matrix $[w_{ij}]$, w_{ij} is the weight between neurons (i,j) and **u**(t) is the output state vector (of dimension N) at time t, given as:

$u_i(t+1) = \mathrm{sgn}(net_i(t)) = 1$ if $net_i(t) \geq 0$,
$= -1$ otherwise, i = 1,2,.. N,

and

$$net_i(t) = \sum_{i=1}^{N} w_{ij}\, u_j(t) + \theta_i \quad (2)$$

where θ_i is the threshold of neuron i.

Letting $u_{N+1} = 1$ and $w_{i,N+1} = \theta_i = w_{N+1,i}$, i = 1,2,...,N, $w_{N+1,N+1} = 0$, the Hopfield network may be considered to have N+1 nodes, with the weights connected to the N+1th node representing the threshold values. Therefore, (1) may be written as :

$$E = -\frac{1}{2}\sum_{i=1}^{N+1}\sum_{j=1}^{N+1} w_{i,j}\, u_i\, u_j \quad (3)$$

$$E = -\frac{1}{2}\, \mathbf{u}^T\, \mathbf{W}\, \mathbf{u}$$

Since **W** is real and symmetric, its eigenvalues are real and its eigenvectors are orthogonal. Let $\lambda_1 \geq \lambda_2 \geq \lambda_3 \ldots\ldots \geq \lambda_{N+1}$ be the eigenvalues of **W** and $e_1, e_2, \ldots, e_{N+1}$ be the corresponding orthonormal eigenvectors. It should be noted that because trace(**W**) = 0, some of the λ_i's are nonnegative and the remaining are negative. Therefore **W** is indefinite and it is possible that $|\lambda_{N+1}| > |\lambda_1|$. An equivalent statement is that although **W** and $\mathbf{W}^2 = \mathbf{W}\mathbf{W}^T$ have the same eigenvectors but

with eigenvalues λ_i and $(\lambda_i)^2$, respectively, they do not necessarily have the same order of the eigenvalues.

The analysis, given in [4], of the dynamics of the discrete Hopfield model, asserts that (3) converges to the nearest hypercube corner in the direction of e_1 (the eigenvector of the connection matrix **W** corresponding to the largest positive eigenvalue). If λ_1 is degenerate, then there exists a corresponding subspace instead of a corresponding eigenvector. In this case, the whole of this subspace is considered as the location of the minimum of (3).

Also, one may note that if **W** is made positive definite (with the same eigenvectors), then it is a standard result [5] that; for a fixed $\|\mathbf{u}\|$ (where $\|\ \|$ denotes the euclidean norm operator), i.e., **u** is constrained to lie on a hypersphere; (3) would be minimized when **u** is along a maximal eigen direction of **W** (or $\mathbf{WW}^T$) . Therefore, we have the following result :

if **W** is made positive definite then the direction that *minimizes* the Hopfield function (3) (with **u** constrained to lie on a hypercube) is the same direction that *maximizes* the following objective functions : (with **u** constrained to lie on a hypershpere):

$$F1 = \mathbf{u}^T \mathbf{W} \mathbf{u}, \qquad (4 - a)$$

$$F2 = \mathbf{u}^T \mathbf{W} \mathbf{W}^T \mathbf{u} \qquad (4 - b)$$

The *objective* of this paper is to solve (3) using (4b) as an energy function for a feedforward network of lower space (or synaptic) complexity than the Hopfield network. Since, currently, the *bottleneck* in designing neural hardware is *the number of connections per unit area*, this line of thought; besides its conceptual value; can lead to practical neural solutions to existing real life problems.

B. Organization of the Paper .

The remaining of the paper is organized as follows : Section I outlines the motivation of the paper and its relation to the **DDHNN**. Section II discusses the preprocessing of **W** to make it positive definite while retaining the same maximal eigen direction. Section III discusses how a simple Oja-like feedforward network will serve our purpose of calculating the maximal eigen direction of the preprocessed Hopfield weight matrix **W**. The complexity; both in time and space; of the new proposed algorithm is estimated. Section IV presents a simple example that illustrates the basic idea proposed in this paper and finally, the paper is concluded in the last section.

I.MOTIVATION BEHIND THE PRESEN WORK.

A..- Informal Discussion :

The initial motivation for the present work was the study of the relation between the **DDHNN** and clustering algorithms (both classical and neural) due to the following observations :

(i) *Intuitively*, it is known that the activity of a **DDHNN** stabilizes with its set of neurons *partitioned* into two classes; each class includes neurons reinforcing each other. This, obviously, corresponds to a (hard) *clustering* process. Clustering is the (*unsupervised*) process of grouping patterns into groups (clusters/classes) according to a certain similarity criterion.

(ii) Most neural networks doing clustering are *single layer feeedforward* networks. It seems natural to inquire whether a feedforward network; that explains the clustering behavior of the **DDHNN**; exists.

(iii) A strong analogy has been shown to exist between annealing and clustering [6] as well as between annealing and the Hopfield model [7].

(iv) An equivalence has been shown to exist between the running of the Minimum Cut Algorithm (MCA) [8] and the running of the **DDHNN**. The operation of the MCA is very similar to that of a clustering algorithm (such as the K-means algorithm [9]), and it is at the origin of the new look, taken in this paper, to the role of both the weight matrix (**W**) and the state vector (**u**) of a **DDHNN**. This is formally discussed in B, below.

B. Formal Discussion :

Mathematically, (using the same notation used in the Introduction the running of the **DDHNN** partitions its set of nodes into two disjoint subsets (called V_1 and V_{-1}) such that : $u_i = 1$ if node $i \in V_1$, and $u_j = -1$ if node $j \in V_{-1}$. Equivalently, we have $\mathbf{u}^T \mathbf{w}_i > 0$ for $i \in V_1$, and $\mathbf{u}^T \mathbf{w}_i < 0$ for $j \in V_{-1}$, ($\mathbf{w}_i$ corresponds to a row in **W**). Therefore, the following objective function is maximized by the running of the **DDHNN** :

$$f(u,W) = \sum_{i=1}^{N+1} \| \mathbf{u}^T \mathbf{w}_i \|^2$$

$$= \sum_{i=1}^{N+1} u^T w_i w_i^T u \qquad (5 - a)$$

$$= u^T (\sum_{i=1}^{N+1} w_i w_i^T) u$$

if W is a real and symmetric matrix, (5a) can be written as

$$= u^T W W^T u \qquad (5 - b)$$

It can be noted that (5-b) has exactly the same form as (4b). This shows how (5b) can be derived, logically, from the **HDDNN** and that it is *equivalent* to it **if W** is a positive definite matrix **and u** is constrained to lie on a hypersphere instead of a hypercube . Moreover **W W**^{T} in (5b) represents a *correlation matrix* and, therefore, is similar to the objective function maximized by the one unit *Oja feedforward network*. This result motivated the remaining of the paper and justifies *the title of the paper*. Provided that **W** is made positive definite, an interesting duality results between the role of each of **W** and **u** in both the feedback (Hopfield) and the feedforward network: the Hopfield weight matrix (**W**) becomes a set of vectors (patterns) that are projected on **u**. In his turn, the vector **u, now,** defines a *hyperplane* (**u** is the normal to this hyperplane) in the *weight space* that is *adapted to act as a two-class classifier* for the weight vectors (the rows of **W**).

II.- TRANSFORMING THE HOPFIELD ENERGY FUNCTION TO A POSITIVE DEFINITE FUNCTION

The idea presented in this section relies on certain properties of the Hopfield model as well as on some well known properties from matrix algebra. First, theorem (1) when associated with Property (1); given below; allow one to increase the diagonal elements of **W** by an amount $\lambda_o > 0$, without changing the nature of the final steady state solution.

Theorem(1) [8] (On the convergence of the discrete Hopfield model):
Let H = (**W**,θ) be a Hopfield neural network. Assume that H is operating in a serial mode and that **W** is a symmetric matrix with the elements of the diagonal being nonnegative. Then the network will always converge to a stable state, that is, there are no cycles in the state space.

Property (1) (Shift of eigenvalues origin):
If λ is an eigenvalue of **A**, then $\lambda + \lambda_o$ is an eigenvalue of $(\mathbf{A} + \lambda_o \mathbf{I})$, where **I** is the identity matrix. The eigenvectors remain the same.
In order to obtain an estimate for λ_o that ensures the positive definiteness of $\mathbf{W} + \lambda_o \mathbf{I}$, theorem(2) is needed,

Theorem (2) [10] If A is a real n*n matrix, then
$[\rho(A^T A)]^{1/2} = \|A\|_2$
where, ?? **(A)** denotes the spectral radius of a matrix **A,** defined as $\rho(\mathbf{A}) = \max |\lambda|$, λ is an eigenvalue of **A** and $\|\mathbf{A}\|$ is the l_2 norm.
Since, in our case, **W** is symmetric, then we have trace $(\mathbf{W}\,\mathbf{W}^T)$ =

$$\sum_{i=1}^{N} \|W_i\|^2 = \sum_{i=1}^{N} \lambda_i^2 > \rho(W)$$

Therefore, taking λ_o= trace $(\mathbf{W}\,\mathbf{W}^T)$ ensures that all the eigenvalues of $(\mathbf{W} + \lambda_o \mathbf{I})$ are positive, i.e. $(\mathbf{W} + \lambda_o \mathbf{I})$ is non-singular.

In the next section, a learning algorithm for a single neuron network is proposed for the extraction of the maximal eigen direction of the positive definite matrix $(\mathbf{W} + \lambda_o \mathbf{I})$, which from the discussion given in the present section, is the same as the maximal eigen direction of **W**.

III- A SINGLE NEURON-BASED LEARNING RULE FOR THE CALCULATION OF THE MAXIMAL EIGEN DIRECTION OF THE HOPFIELD WEIGHT MATRIX

The algorithm proposed in this section is similar to the Oja learning rule [11] for the calculation of the principal component of a random matrix, given as:

$$\mathbf{m}(k+1) = \mathbf{m}(k) + r(k)\, y(k)\, [\mathbf{x}(k) - y(k)\mathbf{m}(k)] \qquad (6)$$

where k is the time step, $\mathbf{x}(k)$ is a sequence of random vectors $\in R^n$; generated from a time-invariant statistical distribution characterized by its mean vector μ and its covariance matrix $\mathbf{C} = \langle (\mathbf{x}-\mu)^T (\mathbf{x}-\mu) \rangle$ (where <.> denotes the expectation operator); $\mathbf{m}(k)$ is the weight vector, of dimension n, of the (linear) neuron whose output (y) is given by: $y(k) = \mathbf{x}^T(k)\, \mathbf{m}(k-1)$ and r(k) is a positive scalar called the learning rate that satisfies:

$$\sum_k r(k) = \infty \,, \sum_k r^2(k) < \infty \,,$$

that is decreased to zero as k approaches ∞

Based on the theory of stochastic approximation, m(k) converges to the first normalized principal component of matrix **C** with probability one as k approaches ∞. For zero mean data, the maximal eigenvector is

equivalent to the principal component of the data. Equation (6) has the advantage of depending on locally available variables for updating of the weight vector. Also, recently in [12], it is proven that there exists an energy function whose steepest descent direction is the same as the average evolution direction of (6). This energy function is of the form

$$F(\mathbf{m}) = \mathbf{m}^T\mathbf{m} - \ln(\mathbf{m}^T\mathbf{C}\mathbf{m}) \qquad (7)$$

and its gradient $\nabla F = 2(\mathbf{m} - \frac{\mathbf{C}\mathbf{m}}{\mathbf{m}^T\mathbf{C}\mathbf{m}})$ (8)

The stationary points of Equation 7 are the normalized eigenvectors $\pm\mathbf{v}_i$ (i=1,2, ..., n) of matrix **C**, with corresponding eigenvalues denoted γ_i. However, only the Hessian matrix $\nabla^2F(\pm\mathbf{v}_1)$ is positive definite, while $\nabla^2F(\pm\mathbf{v}_i)$ is indefinite for i=2,..., n. Therefore, Equation 7 has two global minimal points that correspond to the two converged points of the one unit Oja learning rule and has no other local minimal points.

Noticing that :

$$\langle yx\rangle = \mathbf{C}\mathbf{m}\ ,\ \langle y^2\rangle = \mathbf{m}^T\mathbf{C}\mathbf{m} \qquad (9)$$

and letting the learning rate r(k) in (6) be dependent on the estimated value of γ_1 (as observed in [13]), calculated as

$\gamma_1(k) = \mathbf{m}(k)^T\mathbf{C}\mathbf{m}(k)/\|\mathbf{m}(k)\|^2$, such that,

$$r(k) = \frac{\beta(k)}{\gamma_1(k)} \prec 1 \qquad (10)$$

where $\beta(k)$ is decreased to zero as k approaches infinity, it can be seen that $\Delta\mathbf{m}$ in the Oja rule (6) is proportional to the negative of ∇F (given in (8)).

B.- The New Proposed Algorithm

Let **C** be the (N+1)*(N+1) matrix :

$(\mathbf{W}+\lambda_o\ \mathbf{I})^T(\mathbf{W}+\lambda_o\mathbf{I})$; where the matrix $(\mathbf{W} + \lambda_o\mathbf{I})$ is obtained as explained in section II. One can easily show that the Hessian matrix of the energy function (7) still preserves the same properties with $\nabla^2F(\pm\mathbf{e}_1)$ positive definite and $\nabla^2F(\pm\mathbf{e}_i)$ indefinite for i=2,..., N+1 ($\mathbf{e}_i$ and λ_i denote the eigenvectors and eigenvalues of **W**, respectively).

It is worth noting that choosing $\mathbf{C} = (\mathbf{W} + \lambda_o\mathbf{I})$ is also valid, but the above choice accelerates the convergence of the algorithm proposed below (because the eigenvalues of $(\mathbf{W}+\lambda_o\mathbf{I})^T(\mathbf{W}+\lambda_o\mathbf{I})$ are the squares of those of $(\mathbf{W}+\lambda_o\mathbf{I})$, and, therefore, the discrimination between the maximal eigenvalue and the remaining ones is sharper), and makes better use of the properties (9) of the Oja algorithm. The proposed algorithm updates the weight vector **m** in a deterministic way (i.e, $\beta(k)$ in Equation 10 is set equal to one) and in the negative direction of ∇F (given in (8)), as follows :

$$\mathbf{m}(k+1) = \mathbf{m}(k) + (\frac{1}{\lambda(k)})\,[\mathbf{C}\,\mathbf{m}(k) - \lambda(k)\mathbf{m}(k)] \qquad (11)$$

where, $\lambda(k)$ corresponds to an estimate of the maximal eigenvalue of matrix **C** (i.e., $\lambda=(\lambda_1+\lambda_o)^2$) evaluated as :

$\frac{\mathbf{m}^T(k)\,\mathbf{C}\,\mathbf{m}(k)}{\|\mathbf{m}(k)\|^2}$, $\mathbf{Cm(k)}$ and $\mathbf{m}^T\mathbf{(k)Cm(k)}$ are calculated from the network inputs and outputs as given in (9). It should be noted that since **C** is no longer a covariance matrix, there is no need to calculate the mean of the input vectors, as well as to its substraction from each of them. The new algorithm can be described in pseudocode as follows :

Proposed algorithm :

The algorithm consists of two phases :

(i) a *preprocessing* phase in which a scalar (λ_o) is estimated (as explained in section II), so that $(\mathbf{W}+\lambda_o\mathbf{I})$ becomes a positive definite matrix. In this phase, the rows of **W** are presented, sequentially, to the input of a linear neuron; having a variable weight vector of size (N+1) (i.e. equal to the size of each row in **W**). At each input presentation, the neuron weight vector is set equal to the input vector. Therefore, the output from the neuron gives the square of the norm of the input vector. Obtaining the summation of the output values over the (N+1) input presentations gives the required estimate for λ_o .

(ii) an Oja-like training phase (also using a single linear neuron, as in the previous phase), to calculate the direction of the maximal eigenvector (i.e. the eigenvector corresponding to the maximal eigenvalue) of $(\mathbf{W}+\lambda_o\mathbf{I})^T\ (\mathbf{W}+\lambda_o\mathbf{I})$.

1: Preprocessing of W to calculate λ_o using a single linear neuron

{*Initialize*}

sum ← 0,

For i ← 1 to N+1 do

begin

Present row i of the Hopfield weight matrix **W** to the input of the single neuron network

set **m**← $\mathbf{w}_i$ {*m is the weight vector of the*

neuron}
Calculate the output $y \leftarrow \mathbf{m}^T \mathbf{w}_i$,
sum ← sum + y
end {For}.
Set λ_o ← sum

augment the diagonal elements of **W** by this amount.

2: One unit deterministic Oja network training algorithm

2.1 network initialization

{Initialize}
time counter k ← 0,
the weight vector $\mathbf{m}(0) \in R^{N+1}$ with small random values, the first guess and next guess of the maximal eigenvalue *{called eigen_first and eigen_next* to suitable values to activate the main loop of the algorithm (step 2.2).

2.2 network learning algorithm

{Iterate Until Convergence}
while abs(eigen_first - eigen_next) > ∈ do
{∈is a very small positive number}
begin {while}
eigen_first ← eigen_next,
eigen_next ← 0,
v ← 0 *{**v** a vector of dimension* N+1, *that corresponds to* **Cm** *in* (9), *is initialized to the zero vector}*,

{Read training data. Training is per epoch (i.e. in batch mode) not per sample}
for i ←1 to N+1 do
begin {for i}
Read row i of the Hopfield matrix and present it to the network input as vector $\mathbf{x} \in R^{N+1}$,
Calculate the output $y \leftarrow \mathbf{m}^T \mathbf{x}$,
Update vector v :
for j ←1 to N+1 do $\mathbf{v}[j] \leftarrow \mathbf{v}[j]+y*\mathbf{x}[j]$,
eigen_next ← eigen_next + sqr(y),
end {for i}

{Weight vector Updating}
eigen_next ← eigen_next/ $|\mathbf{m}(k)|^2$;
$\mathbf{m}(k+1) \leftarrow \mathbf{m}(k)+(1/\text{eigen\_next})*(\mathbf{v}-\text{eigen\_next}*\mathbf{m}(k))$,

{Increment the time counter}
k ← k + 1,
end {while}.

C.- Algorithm Complexity

Assuming a *hardware model* composed of a digital RAM of size $O(N^2)$ to store the (N+1)*(N+1) weight matrix **W** and an analog parallel hardware consisting of a single linear neuron with a synaptic weight vector of size O(N) (therefore, network operation is assumed in parallel. Only the presentation of the input vectors is serial), the complexity of the above algorithm; in terms of both storage capacity and execution time; can be estimated as follows :

The Space Complexity for both the preprocessing phase and the network training Phase is the same and is of $O(N^2)$ for the digital RAM and of O(N) for the analog neuron. One may note that, the size of the RAM can be reduced to a minimum if the weights are generated on-line (using the necessary arithmetic operations). This, of course, will increase the time complexity.

The crucial point is the following : because *the current bottleneck in designing neural hardware is the number of connections per unit area*, optimizing the Hopfield objective function using a neural network with O(N) weights in addition to a RAM of $O(N^2)$ is better than designing a purely analog hardware with $O(N^2)$ weights (especially when N gets large).

The Time Complexity :

(i) the Preprocessing Phase : of O(N) to estimate λ_o.

(ii) the Network Training Phase : of O(Nepoch*N), where Nepoch corresponds to the number of passes over the complete training set which includes (N+1) training vectors.

IV- ILLUSTRATIVE EXAMPLE

In this section a simple example is given for the purpose of illustration. However, the objective of this work is to reduce the complexity of the neural network needed to solve optimization problems whose objective functions have a large number of independent variables and that have a one-to-one correspondence with the Hopfield energy function (1).

Consider the following Hopfield Energy function; given in [2]; that describes the design of a 4-bit A/D converter as a simple optimization problem :

$$\max E = -\frac{1}{2}\sum_{j=0}^{3}\sum_{i\neq j=0}^{3}(-2^{(i+j)})\,u_i\,u_j - \sum_{i=0}^{3}(-2^{(2i-1)} + 2^i x)\,u_i \qquad (11\text{-a})$$

where **u** is the output state vector, and x is the analog input. Equation (11a) is of the form (1) if

$$w_{ij} = -2^{(i+j)}, \quad 0 \le i, j \le 3, \; i \neq j, \qquad (11\text{-b})$$

$$\theta_i = (-2^{(2i-1)} + 2^i x), \quad 0 \le i \le 3.$$

Taking x (the analog input), in the range $0 \le x \le 15$, equal to 7 (say), then the HNN has the following parameters:

# of nodes = 5 = 4 + 1 (for the bias node).

Weight Matrix =

0	-2	-4	-8	6.5
-2	0	-8	-16	12
-4	-8	0	-32	20
- 8	-16	-32	0	24
6.5	12	20	24	0

The weight matrix has the following 5 distinct eigenvalues; listed in a descending order 33.53939, 19.05957, 6.431604, 1.377108, -60.40766. The maximal eigenvector $\mathbf{e}_1$ associated with the maximal eigenvalue has been calculated by applying Jacobi's numerical method], and is found to be

$\mathbf{e}_1$=[0.07816665 0.1755662 0.6139752 -0.7591278 -0.09912797]^T

It follows that the final state of the Hopfield network can be obtained by hard limiting each component of the maximal eigenvector. Therefore, the final state vector is found to be u* = sign($\mathbf{e}_1$) = [1 1 1 -1 -1]. Excluding the state assigned to the threshold neuron (the last component of u*) and considering that the first component (the leftmost) corresponds to the least significant bit, the final state vector is the correct bipolar representation of the analog input '7'. It may be worth noting that the vector (-$\mathbf{e}_1$) is also a valid solution that results in an interchange of the set of neurons assigned to '1' with the set of neurons assigned to '-1' [8].

Applying the new technique proposed in section III to solve the same example, the solution proceeds as follows :

1.- Preprocessing : trace($\mathbf{WW}^T$) = 5180.5, therefore, λ_o is selected equal to this number.The diagonal elements of **W** are set to the value of λ_o .

2.- The neural network training algorithm is initialized with , (the tolerance of the stopping criterion)= 0.001, the initial weight vector of the network is set (randomly) to :

m(0) = [0.096 0.117 0.109 0.002 0.153].

The algorithm took 64 training epochs to converge to the following results :

final weight vector :

m* = [0.078 0.176 0.614 -0.759 -0.099]

and its corresponding eigenvalue λ^* = 2.7186e+007. The corresponding eigenvalue of **W** is $\sqrt{\lambda^*} - \lambda_o$ = 33.5394. Comparing these results with those obtained using Jacobi method, it can be concluded that their precision is quite high.

CONCLUSION

This paper presents a new technique to derive *the stable state for an feedback attractor network by a pure feedforward network of lower complexity.* This technique optimizes the Hopfield discrete objective function with a neural (**analog**) hardware having O(N) weights, in addition to a (**digital**) RAM of size O(N^2) (to hold the training patterns), instead of a **purely analog** hardware having O(N^2) weights, at the expense of an increase in the time complexity; as discussed in section III. If the suggested hardware model proves to be feasible then a major *obstacle* to the application of the Hopfield model to real-life optimization problems would be removed.

An *important question* that remains to be answered, is whether the proposed technique can be extended to the continuous (i.e. neurons with sigmoid-like transfer functions and continuous update dynamics [3]) or annealed (continuous stochastic neurons but with discrete update dynamics [5]) Hopfield models. We plan to address this question in the near future. However, it can be readily said that, in general, as long as the continuous neurons operate in their linear parts or in their high gain limits the technique presented in this paper remains applicable.

REFERENCES

[1] J.J. Hopfield, " Neural networks and physical systems with emergent collective computational abilities ", *Proceedings of the National Academy of Science of the USA*, 79, pp. 2445-2558, 1982.

[2] D.W. Tank and J.J. Hopfield, " Simple neural optimization networks : an A/D converter, signal decision circuit and a linear programming circuit ", *IEEE Trans. on Circuits and Systems*, vol. CAS-33, No 3, May 1986.

[3] J.J. Hopfield, " Neurons with graded response have collective computational properties like those of two-state neurons ". *Proceedings of the National Academy of Science of the USA*, 81, pp. 3088-3092, 1984.

[4] S.V. Aiyer et al, " A theoretical investigation into the performance of the Hopfield model ", *IEEE Trans. Neural Networks*, 1(2), pp. 204- 215, 1990.

[5] J. Hertz, A. Krogh and R. Palmer, *Introduction to the Theory of Neural Computation*, Addison- Wesley, 1991.

[6] S. Z. Selim, and K. Alsultan, ," A Simulated Annealing Algorithm for the Clustering problem", *Pattern Recognition*, vol. 24, no 10, pp. 1003-1008-1991.

[7] D.E. van Den Bout and T. K. Miller, " Graph partitioning using annealed networks." IEEE Trans. Neural Networks, vol. 1, no. 2, pp. 192-204, 1990.

[8] J. Bruck, "On the convergence properties of the Hopfield model ", *Proceedings of the IEEE*, vol. 78, No. 10, pp.1579-1584, 1990.

[9] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[10] R.L. Burden, J.D.Faires, A.C. Reynolds, *Numerical Analysis*, Prindle, Weber & Shmidt, USA, 1981.

[11] E. Oja, " A simplified neuron model as a principal component analyzer", *J. Math. Biology*, vol. 15, 1982, pp. 267-273, 1982.

[12] Q. Zhang and Y. Leung, " Energy function for the one-unit Oja algorithm", *IEEE Trans. on Neural Networks*, Vol. 6, No. 5, September 1995, pp. 1291-1293.

[13] L. Chen and S. Chang, " An adaptive learning algorithm for principal component analysis ", *IEEE Trans. on Neural Networks*, Vol. 6, No. 5, September 1995, pp. 1255-1263,1995