International Journal of Computer Applications (0975 – 8887)
Volume 141 – No.5, May 2016

Implementation of Hopfield Neural Network for its Capacity with Finger Print Images

Ramesh Chandra Sahoo
NIET,Gr. Noida (UP)

Somesh Kumar
NIET,Gr. Noida (UP)

Puneet Goswami
NIET,Gr. Noida (UP)

ABSTRACT

This paper analyzes the Hopfield neural network for storage and recall of fingerprint images. The paper first discusses the storage and recall via hebbian learning rule and then the performance enhancement via the pseudo-inverse learning rule. Performance is measured with respect to storage capacity; recall of distorted or noisy patterns. Here we test the accretive behavior of the Hopfield neural network.

Keywords

Hopfield Neural Networks, Associative memory,Pattern storage and recall, Finger print images.

1. INTRODUCTION

Association in human brain refers to the phenomenon of one thought causing us to think of another. Correspondingly, associative memory is the function where the brain is able to store and recall information, given partial knowledge of the information content [1]. Associative Memory is a dynamical system which has a number of stable states with a domain of attraction around them. If the system starts at any state in the domain, it will converge to the locally stable state, which is called an attractor [2]. One such model, describing the organization of neurons in such a way that they function as Associative Memory or also called as Content Addressable Memory, was proposed by J. J. Hopfield and was named after him as Hopfield Model. It is a fully connected neural network model in which patterns can be stored by distributing among neurons and we can retrieve one of the previously presented patterns from an example which is similar to, or a noisy version of it [1, 2]. The network associates each element of a pattern with a binary neuron. The neurons are updated asynchronously and in parallel. They are initialized with an input pattern and the network activation converges to the closest learnt pattern [18]. This dynamical behavior of the neurons strongly depends on the synaptic strength between neurons. The specification of the synaptic strength is conventionally referred to as learning [3]. Learning employs a number of learning algorithms as perceptron, hebbian, pseudo inverse, LMS etc. [32].

The hebbian rule is the simplest rule that can be used to train a network. But it suffers from a number of problems such as:

1. The maximum capacity of this rule is limited to 0.14N, where N is the number of neurons in the network [32].
2. As the number of patterns stored in the network increases, the recall efficiency of the network decreases. [29, 33].
3. The network's ability to correct noisy patterns is also extremely limited and deteriorates with packing density of the network.
4. New patterns could hardly be associated to the stored patterns.

The rule which can be considered to overcome the disadvantages of the hebbian rule is the pseudo inverse learning rule. The pseudo inverse rule is better than the hebbian rule in terms of the capacity, pattern correction and recall efficiency [30, 32]. Section 2 provides a brief description of the Hopfield network as associative memory and its storage and update dynamics. Section 3 elaborates the Pseudo inverse Rule, the associated problems and measures to overcome them. Section 4 contains the experiments whose results have been compiled in Section 5. Discussions and Conclusions then follow in section 6 and 7 respectively

2. HOPFIELD NETWORK AS ASSOCIATIVE MEMORY

Hopfield Network is probably the best known example of a neural network working as associative memory [24, 32]. It is a fully connected network made up of bipolar threshold logic units. The units receive input from every other unit except for itself. The net input of a unit i at any time t is computed by

$$S_i\,(t) = \sum_{i\neq j} W_{ij}\;\; S_j\,(t) \qquad (1)$$

where wij is the weight of the connection between unit i and j and sj is the state of unit j, which can be either +1 or -1 [12, 33].

The next state of unit i is a function of its net input and current state and is given by

$$S_i\,(t+1) = \begin{cases} +1 \; if \; s_i(t) > \theta_i \\ -1 \; if \; s_i(t) \leq \theta_i \end{cases} \qquad (2)$$

θi is assumed to be 0.

The network's weight matrix W is an N × N matrix, whose contents are determined by the set of patterns and the learning rules used to set the weights. The set of stored patterns P={ξ1, ξ2, … ξn) where each pattern ξi is a vector of size n. Thus P is a matrix of size l × n, where l is the number of patterns stored in the network [32 – 34]. The weight matrix in the current paper is a symmetric zero diagonal matrix.

Pattern recall involves setting the initial state of the network equal to an input vector ξi. The states of the individual units are then updated repeatedly until the overall state of the network is stable. Updating of units may be synchronous or asynchronous [1, 22]. In the synchronous update all the units of the network are updated simultaneously and the state of the network is frozen until update is made for all the units. While in the asynchronous update, a unit is selected at random and its state is updated using the current state of the network. This update via random choice of a unit is continued until no further change in the state takes place for all the units i.e. the network reaches a stable state. Each stable state of the network corresponds to a stored pattern that has a minimum hamming distance from the input pattern [11]. With each stable state of the network is associated an energy E and hence that state acts as a point attractor. And during update the network moves from an initial high energy state to the nearest attractor. All stable states which are similar to any of ξi of P are called Fundamental Memories. Apart from them there are other stable states, including inverses of fundamental memories. The number of such fundamental memories and the nature of additional stable states depends upon the learning algorithm that is employed.

3. PSEUDO INVERSE RULE

The pseudo inverse weight matrix is given by

$$W = \Xi\Xi^{-1} \qquad (3)$$

where Ξ is the matrix whose rows are ξn and Ξ -1 is its pseudo inverse. The matrix with the property that Ξ -1 Ξ = I [29, 32, 33].

In contrast to the hebbian rule pseudo inverse rule is neither local nor incremental. This means that to update a particular connection, it does not depend on the information available on either side of the connection and also patterns cannot be incrementally added to the network. These problems can be solved by modifying the rule in such a way that some characteristics of hebbian learning are also incorporated such that locality and incrementally is ensured. The hebbian rule is given as:

$$W_{ij} = \frac{1}{N}\sum_{l=1}^{L} \xi_{li} * \xi_{lj} \quad for\ i \neq j \qquad (4)$$
$$= 0,\ for\ i = j, 1 \leq i,j \leq N$$

where, N is the number of units/neurons in the network ξl for l = 1 to L are the vectors / images to be stored, where each component of ξl is binary i.e. each ξli = ±1 for i=1 to N.

Now the pseudo inverse of the weight matrix can be calculated as

$$W_{pinv} = \mathrm{W^t} *(W * W^t)^{-1} \qquad (5)$$

Where Wt is the transpose of the weight matrix W

(W * Wt)-1 is the inverse of the product of W and its transpose.

This method will overcome the locality and incrementally problems associated with the pseudo inverse rule. In addition it has the benefits of the pseudo inverse rule in terms of the storage capacity and recall efficiency over the hebbian rule.

4. SIMULATION DESIGN AND IMPLEMENTATION DETAILS

Various experiments were conducted to test the efficiency of the pseudo inverse rule for the points mentioned in Section 1. All experiments were conducted in MatLab. Before implementing the learning rule the initial task was conversion of raw fingerprint images into patterns for storage in the network. For this the images were preprocessed through a series of steps, briefly discussed in the following subsection.

4.1.Image Preprocessing

Preprocessing, in the form of image enhancement, of the fingerprint images is required to convert the images into suitable patterns for storage in the Hopfield Network. The term image enhancement refers to making the image clearer for easy further operations. The fingerprint images considered for the study are the images of the fingerprint impressions of different individuals. The images are not of perfect quality to be considered for storage in a network. Hence enhancement methods are required to reveal the fine details of the images which may remain uncovered due to insufficient ink or imperfect impressions. The enhancement methods would increase the contrast between image components and connect the broken or incomplete image components.

The images were first scanned as RGB images and then converted to Grayscale to retain the fine details in the images. Then the image was enhanced and made clearer and sharper using histogram equalization techniques. Note that histogram equalization refers to expansion of the pixel value distribution of an image so as to increase the perceptional information. The image was then subjected to binarization. Binarization refers to conversion of a grayscale image to black and white image. Typically binarization converts an image of upto 256 gray levels to a black and white image.

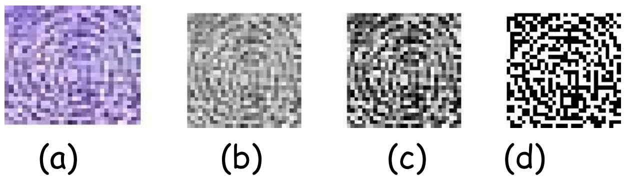

Fig 1: (a) Initial RGB Image, (b) Grayscale Image, (c)Histogram equalized Image, (d) Binarized Image

After attaining binarization, the need was to convert the binary image to bipolar image, since Hopfield networks work best with bipolar data. A bipolar image is one where each pixel has value either +1 or -1. Hence, all pixel values are verified and those with value 0 are converted to -1, thus converting the image to bipolar. Finally the image is converted to bipolar vectors. All the image vectors are stored into a comprehensive matrix which has the following structure.

$$P = \begin{pmatrix} \xi_{11}, \xi_{12}, \ldots\ldots\ldots, \xi_{1N} \\ \xi_{21}, \xi_{22}, \ldots\ldots\ldots, \xi_{2N} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \xi_{11}, \xi_{12}, \ldots\ldots\ldots, \xi_{1N} \end{pmatrix}$$

4.2.Pattern Storage

The patterns in the form of bipolar vectors created in section 4.1 were then stored in the Hopfield network via the following algorithm separately for hebbian and pseudo inverse rules in separate weight matrices.

Algorithm: Pattern Storage

1. Begin with a zero weight matrix, W of size N × N.
2. Compute the weight matrix as per equation 4 for the l patterns, where each pattern is a vector of size n.
3. Assign zeros to the diagonal elements at the end of the each iteration.
4. Finally calculate the pseudo inverse of W as per equation 5.

Hence the L patterns are all stored in the Hopfield network. The final weight matrix is a symmetric zero diagonal matrix. Once the patterns were stored the next experiments were to test the capacity and recall efficiency of the network. This is discussed in the next subsection.

4.3.Pattern Recall

4.3.1. Mathematical Implementation

Pattern recall refers to the identification and retrieval of the corresponding image when an image is presented as input to the network. As soon as an image is fed as input to the network, the network starts updating itself. In the current paper, we use asynchronous update of the network units to find their new states. This update via random choice of a unit is continued until no further change in the state takes place for all the units. That is, the state at time (t+1) is the same as the state at time t for all the units.

$$s_i(t+1) = s_i(t)\ for\ all\ i \qquad (6)$$

Such a state is referred to as the stable state. In a stable state the output of the network will be a stable (trained) pattern that has a minimum hamming distance from the input pattern [11]. The network is said to have converged and recalled the pattern if the output matches the pattern presented initially as input. For pattern association, the patterns stored in an associative memory act as attractors and the largest hamming distance within which almost all states flow to the pattern is defined as the radius of the basin of attraction [5].

Each state of the Hopfield Network is associated with an energy value, whose value either reduces or remains the same as the state of the network changes [22]. The energy function of the network is given by

$$V = -\frac{1}{2}\sum_i \sum_j w_{ij} s_i s_j + \theta_i s_i \qquad (7)$$

Hopfield has shown that for symmetric weights with no self feedback connections and bipolar output functions, the dynamics of the network using asynchronous update always lead towards energy minima at equilibrium. The network states corresponding to these energy minima are termed as stable states [22] and the network uses each of these stable states for storing individual patterns.

The Hopfield network has the ability to recognize unclear pictures correctly. This means that the network can recall actual pattern when the noisy or partial clues of that pattern are presented to the network. It is known and has been shown [1, 2] that Hopfield networks converge to the original patterns if up to 40% distorted version of a stored pattern is presented. The patterns are stored in the network in the form of attractors on the energy surface. The network can then be presented with either a portion of one of the images (partial cue) or an image degraded with noise (noisy cue) and through multiple iterations it will attempt to reconstruct one of the stored images.

4.3.2. Algorithmic Implementation

The algorithm for pattern recall in a Hopfield Neural Network storing L patterns is as follows:

The algorithm would be implemented both for Hebbian and Pseudo inverse rules and results would be recorded. Assume a pattern x, of size N, already stored in the network and modified by altering the values of k randomly chosen pixels. Also, assume vectors ynew to store the new states of the network. Consider a variable count initialized to value 1.

1) Initialize weights to store patterns (Use Hebbian and Pseudo inverse Rule) as per algorithm for Pattern Storage. While activations of the net are converged perform steps 2 to 8.

2) For each input vector x, repeat steps 3 to 7.

3) Set initial activations of the net equal to the external input vector x, $y_i = x_i$ (i=1 to n).

4) Perform steps 5 to 7 for each unit y_i.

5) Compute the net input $Y_{new} = x_i + \sum_{ij} y_i * w_{ji}$

6) Determine the activation (output signal)

$$y_i = \begin{cases} +1 & if\ y_{new} > 0 \\ -1 & if\ y_{new} \le 0 \end{cases}$$

7) Broadcast the value of y_{new} to all other units.

8) Test for convergence as per equation 4.

Note: The value of threshold θ is assumed to be zero. Each unit is randomly chosen for update.

The maximum number of patterns successfully recalled by the above procedure is a pointer to the maximum storage capacity of the Hopfield Network, which is further discussed in the results. Further the recall efficiency for noisy patterns is also determined as up to what percentage of error in the patterns is acceptable by the network and convergence to the original pattern occurs. This is discussed in the results.

5. RESULTS

1) Experiment 1 (Pattern Recall ability and Critical Storage Capacity of the Hopfield Network). Table 1 shows the maximum number of patterns that can be efficiently stored in the Hopfield network with hebb and pseudo inverse rule.

2) Experiment 2 (Recall Efficiency of the Hopfield Network for Noisy Patterns and New Patterns). Table 2 compares for hebb and pseudo inverse rule, the successful and unsuccessful recall results at 30%, 40%, 50% and 60% distortion respectively at various packing densities of the network. The tables also shows the results that when new patterns were presented to the network, whether they were associated to some stored pattern that matched it most closely or not.

6. DISCUSSIONS

6.1.Pattern Storage

All the patterns after preprocessing as depicted in Section 4.1 were converted into bipolar vectors ready for storage into the Network. As per the algorithm of 4.2, the patterns were input one by one into the Hopfield Network first by hebbian rule and then by pseudo inverse rule. The weight matrices for both are 900 × 900 zero diagonal, symmetric matrix.

6.2.Pattern Recall and Critical Storage Capacity

The storage capacity of a neural network refers to the maximum number of patterns that can be stored and successfully recalled for a given number of nodes, N. The Hopfield network is limited in storage capacity to 0.14N when trained with hebbian rule [1-5, 11-12, 18, 21]. But the capacity enhances to N with pseudo inverse rule. Experiments were conducted to check the same and the network was able to store and perfectly recall 0.14N i.e. 126 patterns with hebbian rule and N i.e. 900 patterns with pseudo inverse rule.

Thus the critical storage capacity for the Hopfield Network comes out to be 0.14N with hebbian and N with pseudo inverse rule without any error in pattern recall, for the current study.

Table 1: Critical Storage Capacity

	Hebbian	**Pseudo inverse**
Capacity	126	900

6.3.Recall of Noisy Patterns

The Hopfield network has the ability to recognize unclear pictures correctly. This capacity of the Network was tested as per algorithm in Section IV part C. This time the experiments were conducted with the Network storing 20, 50, 80, 100, 130 patterns successively. The patterns were modified by altering 30%, 40%, 50% and 60% bits. The bits were randomly selected for alteration and the results compiled. Table 2 displays the results for both rules at various packing densities of the network and various levels of distortions. Further the network's behavior is also tested for new patterns and the observations are recorded in the same table.

Table 2: Pattern Recall Ability With Noisy And New Patterns

No. of Patterns in the Network	**Pattern Associated to**		
	%distortion	**Hebbian**	**Pseudo inverse**
20 patterns	30%	Original Pattern	Original Pattern
	40%	Original Pattern	Original Pattern
	50%	Nearest neighbor	Nearest neighbor
	60%	None	None
	New Patterns	None	Nearest neighbor
50 patterns	30%	Original Pattern	Original Pattern
	40%	Original Pattern	Original Pattern
	50%	None	Nearest neighbor
	60%	None	None
	New Patterns	None	Nearest neighbor
80 patterns	30%	Original for 50% patterns	Original Pattern
	40%	None	Original Pattern
	50%	None	None
	60%	None	None
	New Patterns	None	None
100 patterns	30%	Original for 30% patterns	Original Pattern
	40%	None	Original Pattern
	50%	None	None
	60%	None	None
	New Patterns	None	None
130 patterns	30%	Original for 10% patterns	Original Pattern
	40%	None	None
	50%	None	None
	60%	None	None
	New Patterns	None	None

Observations

1) With 20 patterns, behavior with distorted patterns is similar with both the rules i.e. up to 40% distortion the same pattern is associated but at 50% distortion some other stored pattern or the nearest neighbor is associated. New patterns are not recognized by the hebbian rule but by pseudo inverse they are associated to some stored pattern.

2) With 50 patterns, behavior up to 40% distorted patterns is similar with both the rules. But at 50% distortion hebb rule a stop recognizing the pattern while pseudoinverse associates it to its nearest neighbor. New patterns are not recognized by the hebbian rule but by pseudo inverse they are associated to some stored pattern.

3) With 80 patterns hebb rule further deteriorates and even at 30% distortion recognizes only 50% patterns. The pseudo inverse rule on the other hand recognizes the patterns up to 40% distortion also but now it stops associating new patterns to stored patterns.

4) With 100 patterns, hebb rule further deteriorates and recognizes only 30% patterns at 30% distortion. While pseudo inverse has the same behavior as that with 80 patterns.

5) With 130 patterns, hebb rule recognizes hardly 10% patterns at 30% distortion. While pseudo inverse rule, though having degraded in performance at this stage yet is able to recognize the original pattern up to 30% distortion. Beyond this there is no recognition.

It was observed that the network performs sufficiently well with pseudo inverse rule than with hebbian rule with regard to distorted patterns and with new patterns which are not at all recognized bt the network in case of hebbian rule.

Further it has been observed that the network's efficiency starts deteriorating as the network gets saturated. As depicted in table 2, the performance of the network deteriorated with 80 patterns for hebb rule and 130 patterns for pseudo inverse rule. This result can be attributed to the reduction of the Hamming Distance between the stored patterns and the consequent reduction of the radius of the basin of attraction of each stable state. Hence only few patterns could settle into the stable states of their original patterns.

7. CONCLUSIONS

The Hopfield network designed to work with image patterns works variably with the learning rules utilized to train the network. The efficiency is invariably low when we work with the hebbian rule in terms of the maximum capacity and the recall efficiency with noisy and new patterns. But this capability can be sufficiently enhanced when the pseudo inverse rule with capabilities of hebbian rule is used to train the network. With this the capacity suddenly increases to N and the recall efficiency is also remarkably improved for distorted patterns. New patterns are now recognized by the network upto a limited packing density which was totally absent with hebb rule.

8. REFERENCES

[1] Kevin Takasaki, "Critical Capacity of Hopfield Networks", MIT Department of Physics, 2007.

[2] Ramasamy Ramachandran, Natarajan Gunusekharan, "Optimal Implementation of two Dimensional Bipolar Hopfield Model Neural Network", Proc. Natl. Sci. Counc. ROC(A), 2000, Vol. 24(1), pp 73 – 78.

[3] Gang Wei, Zheyuan Yu, "Storage Capacity of Letter Recognition in Hopfield Networks", Faculty of Computer Science, Dalhousie University, http://citeseer.ist.psu.edu/584397.html

[4] J.Ma, "The Object Perceptron Learning Algorithm on Generalised Hopfield Networks for Associative Memory", Neural Computing and Applications, 1999, Vol. 8, 25 – 32.

[5] N. Davey, S.P Hunt, "The Capacity and Attractor Basins of Associative Memory Models".

[6] Sylvain Chartier, Richard Lepage, "Learning and Extracting Edges from Images by a Modified Hopfield Neural Network", 2002.

[7] Mahmoud I. A. Abdulla, Hanaa Shaker, "Fingerprint Matching Techniques using Wavelet Analysis".

[8] Marjory Abreu, Michael Fairhurst, "An Empirical Comparison of Individual Machine Learning Techniques in Signature and Fingerprint Classification".

[9] Atsushi Sugiura, Yoshiyuki Koseki, "A User Interface Using Fingerprint Recognition – Holding Commands and Data

Objects on Fingers", UIST '98, 1998, ACM, 0-58113-0341/98/11.

[10] Sung Bum Pan, Youn Hee Gil, Daesung Moon, Yongwha Chung, Chee Hang Park, "A Memory Efficient Fingerprint Verification Algorithm Using a Multi-Resolution Accumulator Array", ETRI Journal, 2003, Vol. 25(3).

[11] E. Vonk, L.P.J Veelenturf, L.C. Jain, "Neural Networks: Implementations and Applications", IEEE AES Magazine, 1996.

[12] Stainslaw Jankowski, Andrzej Lozowski, Jacek M. Zurada, "Complex Valued Multistate Neural Associative Memory, IEEE Transactions on Neural Networks, 1996, 7(4), 1491 – 1496.

[13] Yingquan Wu, Dimitris A. Pados, "A Feedforward Bidirectional Associative Memory", IEEE Transactions on Neural Networks, 2000, Vol. 100(40), 859 – 866.

[14] Ju Cheng Yang, Sook Yoon, Dong Sun Park, "Applying Learning Vector Quantization Neural Network for Fingerprint Matching", AI, 2006, LNAI 4304, 500 – 509.

[15] Ali Okatan, Cagatay Akpolat, Servet Senyucel, "Cosine Transform Method for Fingerprint Recognition".

[16] Haijun Zhou, Reinhard Lipowsky, "Dynamic Pattern Evolution on Scale Free Networks", PNAS, 2005, Vol.102(29), 10052 –10057.

[17] Jianjiang Feng, Zhengyu Ouyang, Anni Cai, "Fingerprint Matching Using Ridges", Pattern Recognition, 2006, Vol. 39, 2131 – 2140.

[18] Amos Storkey, "Increasing the Capacity of a Hopfield Network Without Sacrificing Functionality", Neural Systems Group, London.

[19] Sung Jung Hsiao, Kuo Chin Fan, Wen Tsai Sung, Shih ChingOu, "Innovative Algorithms for Running a Web-Based Pattern Recognition Search System for a Component Patterns Database", Malaysian Journal of Computer Science, 2002, Vol. 15(2), 78 – 93.

[20] Colin Molter, Utku Salihoglu, Hugues Bersini, "Introduction of a Hebbian Unsupervised Learning Algorithm to Boost the Encoding Capacity of Hopfield Networks".

[21] Agnes Meyder, Constantin Kiderlen, "Fundamental Properties of Hopfield NBetworks and Boltzmann Machines for Associative Memories", Machine Learning, vt 2008.

[22] B. Yegnanarayana, Artificial Neural Networks.

[23] Frank Emmert Streib, "Active Learning in Recurrent Neural Networks Facilitated by a Hebb-like Learning Rule with Memory", Neural Information Processing – Letters and Reviews, November 2005, Vol. 9, No. 2, pp 31 – 40.

[24] Dmitry O. Gorodnichy, "The Influence of Self Connection on the Performance of Pseudoinverse Autoassociative Networks".

[25] N. Davey, R.G.Adams, "Stochastic Dynamics and High Capacity Associative Memories".

[26] M.Brown, J.Austin, "Invariant Pattern Recognition Using Binary Neural Networks".

[27] D.M. Titterington, "Bayesian Methods for Neural Networks and Related Models", Statistical Science, 2004, Vol. 19, No. 1, 128– 139.

[28] J. Schmidhuber, "Reducing the Ratio Between Learning Complexity and the Number of Time Varyning Variablesin Fully Recurrent Nets", In Proceedings of the Internationla Conference on

Artificial Neural Networks, Amsterdam, pp 460 – 463. Springer, 1993.

[29] L.F.Abbott, Yair Arian, "Storage Capacity of Generalized Networks", Rapid Communications, Physical Review A, November 15, 1987, Vol. 36, No. 10, pp 5091 – 5094.

[30] W.Tarkowski, M.Lewenstein, A.Nowak, "Optimal Architectures for Storage of Spatially Correlated Data in Neural Network Memories", ACTA Physica Polonica B, 1997, Vol. 28, No.7, pp 1695 – 1705.

[31] Christophe L. Labiouse, Albert A. Salah, Irina Starikova, "The Impact of Connectivity on the Memory Capacity and the Retrieval Dynamics of Hopfield –type Networks.

[32] G.Atithan, "A Comparative Study of Two Learning rules for Associative Memory", PRAMANA – Journal of Physics, December 1995, Vol. 45, No. 6, pp 569 – 582.

[33] C.J.Perez Vicente, "Hierarchical Neural Network with High Storage Capacity", Physical Review A, November 1989, Vol. 40, No.9, pp 5356 – 5360.

[34] J.J.Hopfield, "Neural Networks and Physical Systems with emergent Collective Computational Abilities", PNAS, 1982, Vol. 79, pp 2554 -2558.

[35] L.Personnaz, I.Guyon, G.Drefus, "Collective Computational Properties of Neural Networks: New Learning Mechanisms", Physical Review A, 1986, Vol. 34, No. 5, pp 4217- 4228