Enhancing the Delta Training Rule for a Single Layer Feedforward Heteroassociative Memory Neural Network

Omar Waleed Abdulwahhab
University of Baghdad
College of Engineering
Computer Engineering Department

ABSTRACT
In this paper, an algorithm is suggested to train a single layer feedforward neural network to function as a heteroassociative memory. This algorithm enhances the ability of the memory to recall the stored patterns when partially described noisy inputs patterns are presented. The algorithm relies on adapting the standard delta rule by introducing new terms, first order term and second order term to it. Results show that the heteroassociative neural network trained with this algorithm perfectly recalls the desired stored pattern when 1.6% and 3.2% special partially described noisy inputs patterns are presented.

General terms
Soft computing

Keywords
Associative memory, neural network, partially described input patterns, delta adaptation rule.

1. INTRODUCTION
Associative memories belong to a class of neural networks that learns according to a certain recording algorithm. They usually acquire information a priori, and their connectivity (weight) matrices most often need to be formed in advance. Writing into memory produces changes in the neural interconnections [1].
The memory should have as large a capacity as possible or a large *P* value, which denotes the number of stored prototypes. At the same time the memory should be able to store data in a robust manner, so that local damage to its structure does not cause total breakdown and inability to recall. In addition, the ideal memory should truly associate or regenerate stored pattern vectors and do so by means of specific similarity criteria. Another very desirable feature of memory would be its ability to add and eliminate associations as storage requirements change [1].
Selection of correct weights and thresholds for the neural network is very important for solving pattern recognition and association problems for a given set of input/output pairs [2].
Associative memories also provide one approach to the computer engineering problem of storing and retrieving data based on content rather than storage address. Since information storage in a neural net is distributed throughout the system (in the net's weights), a pattern does not have a storage address in the same sense that it would if it were stored in a traditional computer [3].
Noisy inputs can be classified mainly into two types: partial description of original patterns and distorted version of original pattern [4]. A novel associative memory that performs well in online incremental learning is proposed such that it is robust to noisy data because noisy associative patterns are presented sequentially in a real environment [4].
Some authors have studied reducing the effect of noise for both recurrent and multilayer backpropogation neural networks [5].
Discussions of various neural network algorithms and comparisons of the algorithms have been performed in [6], based on noise in weights, noise in inputs, loss of connections, and missing information and adding information. In [7] a theoretical justification of the fact that kernel vectors completely fails when a particular kind of erosive noise, even of very low percentage, corrupts the input pattern. A new method is proposed for the construction of kernel vectors for binary patterns associations. A new associative memory to improve its noise tolerance and storage capacity is proposed, which is an improved multidirectional associative memory that uses autoassociative bottleneck neural networks to remove noise in its input [8]. A study of Hopfield model of associative memory and the problems of apparition of spurious patterns in the learning phase and its reduced capacity and presented a method to avoid spurious patterns is done by [9]. A derivation of the uncertainty of the associative memory that consists of all the binary vectors with an arbitrary number of input words is given in [10]. Ref. [11] showed how iterative retrieval strategies emerge naturally from considerations of probabilistic inference under conditions of noisy and partial input and a corrupted weight matrix. A description of the development of neural network models for noise reduction is given in [12]. The networks are used to enhance the performance of modeling captured signals by reducing the effect of noise. Ref. [13] presented an algorithm, called ANR (automatic noise reduction), as a filtering mechanism to identify and remove noisy data items whose classes have been mislabeled. A solution have been proposed by [14] to the problem of bias errors on the predicted output when noisy input data is used. A better approximation of a function by combining neural networks with noise reduction algorithms is achieved in [15]. An extensive use of minimax algebra is made by [16] to analyze gray-scale autoassciative morphological memories (AMM) and provided a complete characterization of the fixed points an basins of attractions to describe the storage and recall mechanisms of gray-scale AMMs.
The associative memory used in this paper is a single layer feedforward neural network. The proposed algorithm enhances the robustness of this memory so that when a partially described pattern due to a local damage to its structure is presented, the network can recall it perfectly. This algorithm does not require any modification to the architecture of the network. However, while this algorithm enhances the operation of the network, it comes with the expense of increasing the number of epochs required to reach a zero squared errors. This is an expected fact since this algorithm adds an extra term(s) to the standard delta learning rule, as will be shown later.

2. DIFFERENTIATION PATTERNS
Differentiation patterns are patterns that differentiate between the original input patterns. In this paper, only first order and second

order differentiation patterns are defined. Higher order differentiation patterns can be defined with some extension. To evaluate these differentiation patterns, an accumulation function of the input patterns is defined as

$$f(m) = \sum_{p=1}^{m} x^p$$

where x^p is the pth training pattern. Consider the ith component of f, designated by f_i . For $1 \le m \le P$

$$f_i(m) = \sum_{p=1}^{m} x_i^p$$

where x_i^p is the ith component of x^p. For $m = 1$, if $f_i(1) = 1$, then $p_{i,1} = 1$. For each m, where $1 < m \le P$, if $f_i(m) > f_i(m-1)$, then $p_{i,j} = m$, where $1 < j \le P$. Since

$$f_i(P) = \sum_{p=1}^{P} x_i^p = -n + 2N_{on}$$

where N_{on} is the number of patterns that have f_i=1, therefore, if $f_i(P) = -n + 2$, then only one pattern $p_{i,1}$ has f_i=1. This means that if the input corresponding to this component is on, it will distinguish pattern $p_{i,1}$ from other patterns. Similarly, if $f_i(P) = -n + 4$, then only two patterns $p_{i,1}$ and $p_{i,2}$ have f_i=1. This means that if the input corresponding to this component is on, it will distinguish patterns $p_{i,1}$ and $p_{i,2}$ from the remaining patterns, but without differentiation between $p_{i,1}$ and $p_{i,2}$. Similar conclusions can be drawn for larger values of $f_i(P)$.

2.1 First Order Differentiation Patterns

For each pattern p, collect the components that has $f_i(P) = -n + 2$ and $p_{i,1} = p$. Thus, the differentiating vector for pattern p can be defined as

$diff_p = [diff_i]$, where

$$diff_i = \begin{cases} 1 \text{ if } f_i(P) = -n + 2 \text{ and } p_{i,1} = p \\ 0 \qquad\qquad\qquad \text{otherwise} \end{cases}$$

2.2 Second Order Differentiation Patterns

For each two patterns p_s and p_t, collect the components that has $f_i(P) = -n + 4$, $p_{i,1} = p_s$, and $p_{i,2} = p_t$ Thus, the differentiating vector for patterns p_s and p_t can be defined as

$diff_{p_s p_t} = [diff_i]$, where

$$diff_i = \begin{cases} 1 \text{ if } f_i(P) = -n + 4, p_{i,1} = p_s, p_{i,2} = p_t \\ 0 \qquad\qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

If the desired output of a certain neuron is 1 for the two patterns, the weights connecting this neuron with the input corresponding to the collected components are increased, while if this output is -1, these weights are decreased. On the other hand, if this output is different for the two patterns, these weights are not changed. This can be achieved by adding a scaled outer product of $d^s + d^t$ and $diff_{p_s p_t}$ to the weight matrix of the neural network.

3. PROPOSED ALGORITHM

The standard delta adaptation rule when the activation function is linear is

$$W(t+1) = W(t) + \gamma\big(d^p - o^p(t)\big)x^p$$

The first order term associated with an input pattern is defined as the scaled outer product of the desired output pattern with the first order differentiation pattern, both associated with this input pattern. Thus,

$$F.O.T.(p) = \gamma_P d^P diff_p^T$$

$$W(t+1) = W(t) + \gamma\big(d^p - o^p(t)\big)x^p + \gamma_p d^p diff_p^T$$

For each epoch, the result will be the addition of the sum of all first order terms to the final weight matrix. This sum is

$$\gamma_1 d^1 diff_1^T + \gamma_2 d^2 diff_2^T + \cdots + \gamma_P d^P diff_p^T = \sum_{p=1}^{P} \gamma_p d^p diff_p^T$$

The second order term associated with two input patterns is defined as the scaled outer product of the sum of the desired output patterns with the second order differentiation pattern, both associated with these input patterns. Thus,

$$S.O.T.(p_s,p_t) = \gamma_{p_s p_t}(d^s + d^t) diff_{p_s p_t}^T$$

$$W(t+1) = W(t) + \gamma\big(d^p - o^p(t)\big)x^p + \gamma_p d^p diff_p^T + \sum_{t=p+1}^{P} \gamma_{pt}(d^p + d^t) diff_{pt}^T$$

For each epoch, the result will be the addition of the sum of all second order terms to the final weight matrix. This sum is

$$\gamma_{12}(d^1 + d^2) diff_{12}^T + \gamma_{13}(d^1 + d^3) diff_{13}^T + \cdots + \gamma_{1P}(d^1 + d^P) diff_{1P}^T + \gamma_{23}(d^2 + d^3) diff_{23}^T + \gamma_{24}(d^2 + d^4) diff_{24}^T + \cdots + \gamma_{2P}(d^2 + d^P) diff_{2P}^T + \cdots + \gamma_{(P-1)P}(d^{P-1} + d^P) diff_{(P-1)P}^T$$

$$= \sum_{s=1}^{P-1} \sum_{t=s+1}^{P} \gamma_{st}(d^s + d^t) diff_{st}^T$$

The proposed algorithm is implemented on a heteroassociative neural network shown in Fig. 1. The network associates the three letters A, B, and C from different fonts [4]. The input vector has 63 components and the output vector has 15 components. The activation function of the output neurons is linear. Fig. 2 shows the input characters and their associated output characters. In [4], only distorted input patterns are considered that have 30% noise. Partially described input patterns are not considered there.

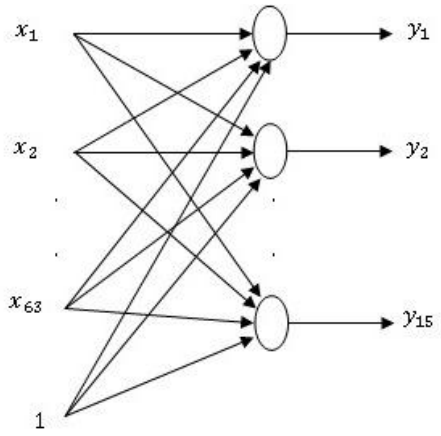

Fig. 1 The architecture of the neural network

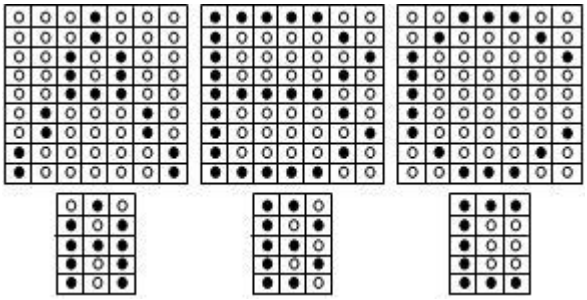

Fig. 2 Input-output character association

In the vector representation of the input and output figures, the black circle is represented by 1 and the white circle by -1. A component that has a value of 0 represents an unsure or missing data. The first order differentiation patterns for the input patterns are

$\boldsymbol{diff}_A = [diff_i]$, where

$$diff_i = \begin{cases} 1 & \text{for } i = 11, 17, 19, 24, 26, 37, 44, 48, \\ & \quad 56, 63. \\ 0 & \text{Otherwise} \end{cases}$$

$\boldsymbol{diff}_B = [diff_i]$, where

$$diff_i = \begin{cases} 1 & \text{for } i = 1, 2, 8, 27, 30, 58. \\ 0 & \text{Otherwise} \end{cases}$$

$\boldsymbol{diff}_C = [diff_i]$, where $\quad diff_i = \begin{cases} 1 & \text{for } i = 9, 51. \\ 0 & \text{Otherwise} \end{cases}$

The second order differentiation patterns are

$\boldsymbol{diff}_{A,B} = [diff_i]$, where

$$diff_i = \begin{cases} 1 & \text{for } i = 31, 32, 33, 41, 50, 57. \\ 0 & \text{Otherwise} \end{cases}$$

$\boldsymbol{diff}_{A,C} = [diff_i]$, where $\quad diff_i = 0$ For all $1 \le i \le 63$

$\boldsymbol{diff}_{B,C} = [diff_i]$, where $\quad diff_i$

$$= \begin{cases} 1 & \text{for } i = 3, 5, 13, 15, 21, 22, 29, 36, 43, \\ & \quad 49, 55, 59, 60, 61. \\ 0 & \text{Otherwise} \end{cases}$$

4. RESULTS AND DISCUSSION

4.1 Standard Delta Adaptation

The neural network was trained using the standard delta rule with zero initial weights and a stopping criteria of zero cumulative error, i.e.,

$$E = \frac{1}{2}\sum_{p=1}^{3}\sum_{k=1}^{15}(d_{pk} - y_{pk})^2 = 0$$

To analyze the operation of the neural network effectively, an input pattern with zero components is applied to inspect the effect of the biases. The output is shown in Fig. 3.

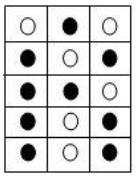

Fig. 3 Output due to biases only

An input pattern with only one correct component that differentiates character **A** from other characters is applied, i.e., 1.6% partially described pattern. Fig. 4 shows the input-output associations. The output is correct due to the effect of the biases. The same output results for inputs with different correct components for the character **A**.

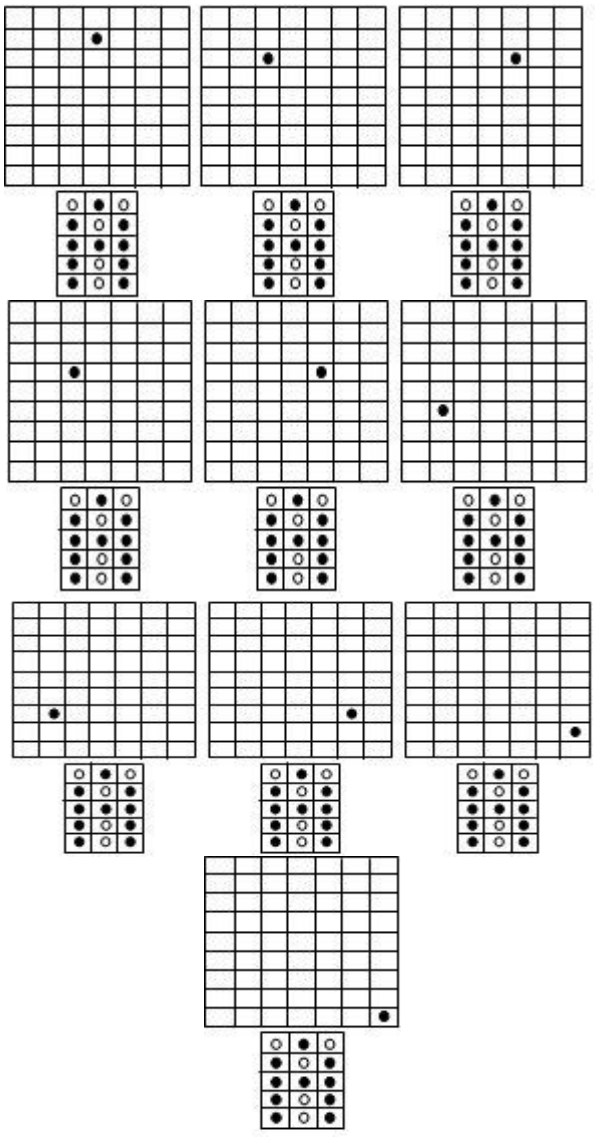

Fig. 4 Input-output associations for 1.6% partially described character A

However, an input pattern with only one correct component that differentiates character **B** from the other characters is applied. Fig. 5 shows the input-output associations. The output is not the desired character **B**. The same thing is true for character **C**, as shown in Fig. 6. This means that the standard delta adaptation rule fails to recall the desired output for a 1.6% partially described pattern for characters **B** and **C**.

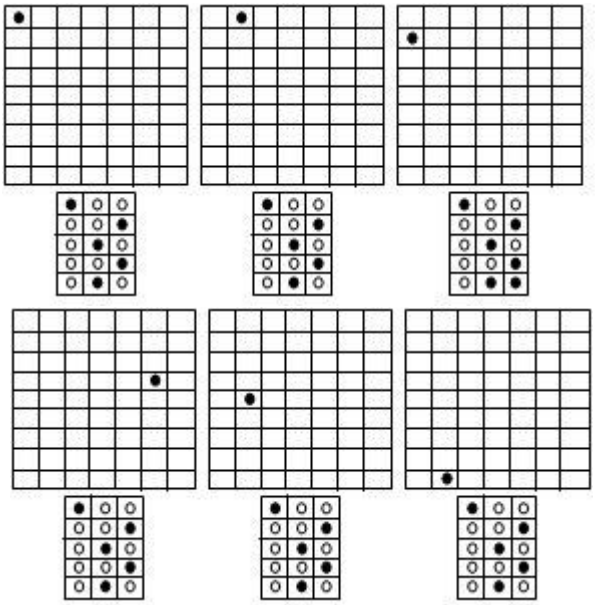

Fig. 5 Input-output associations for 1.6% partially described character B

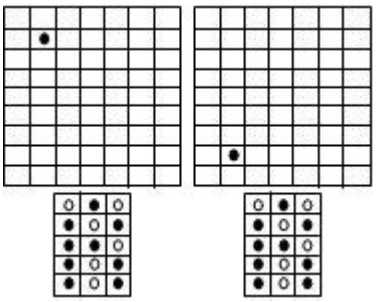

Fig. 6 Input-output associations for 1.6% partially described character C

4.2 Adding the First Order Term

After adding the first order term to the delta adaptation rule, the same 1.6% partially described input patterns are applied to the neural network for differentiating characters **A**, **B**, and **C**. Fig. 7 and Fig. 8 show the input-output associations for characters **B**, and **C**, respectively. Now, the neural network perfectly recalls the desired output for character **B** and **C**. The input-output associations for character A is the same as that shown in Fig. 4.

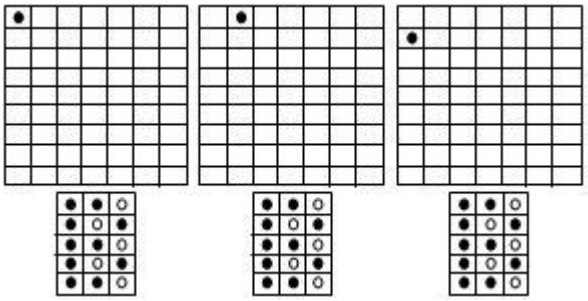

Fig. 7 Input-output associations for 1.6% partially described character B with first order term added

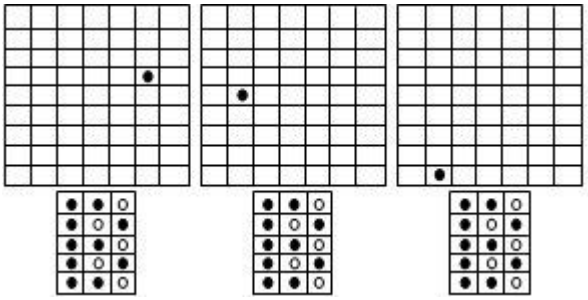

Fig. 7 Contd.

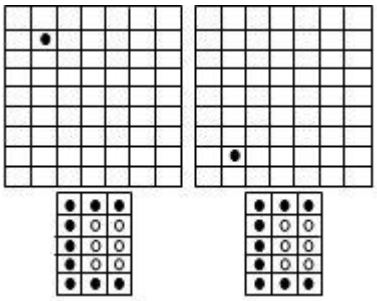

Fig. 8 Input-output associations for 1.6% partially described character C with first order term added

Before adding the second order term, a 3.2 % partially described input pattern with only two correct components, one differentiates characters **A** and **B** from character **C** and the other differentiates characters **B** and **C** from character **A**, is applied. Considering all the possible combinations of such components, there are $6 \times 14 = 84$ different input patterns. All these patterns are 3.2% partially described. The desired output of all these input patterns is the common character, which is character **B**. However, the actual output of each input pattern is not character **B**. Because of their large number (84), an arbitrary selected six of them are shown in Fig. 9.

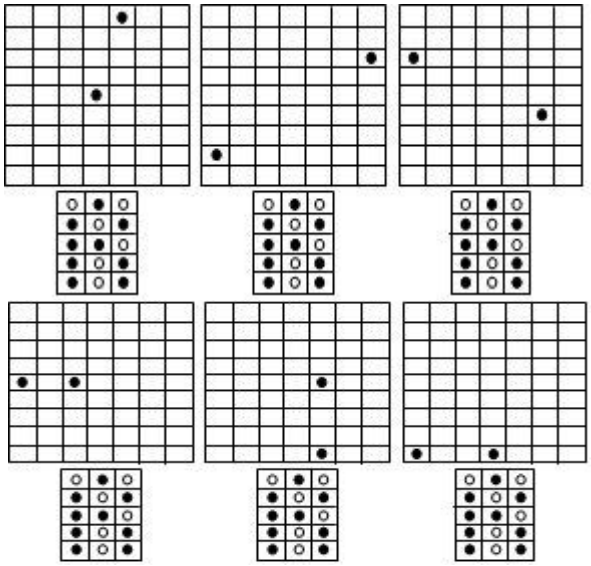

Fig. 9 Input-output associations for 3.2% partially described input pattern

4.3 Adding the Second Order Term

After adding the second order term, the same 3.2 % partially described input patterns are presented to the neural network. The actual output of *all* these input patterns is the desired output, character **B**. As shown in Fig. 10, in all cases the neural network perfectly recalls character **B**. Also, because of their large number, the same previously selected six patterns are shown.

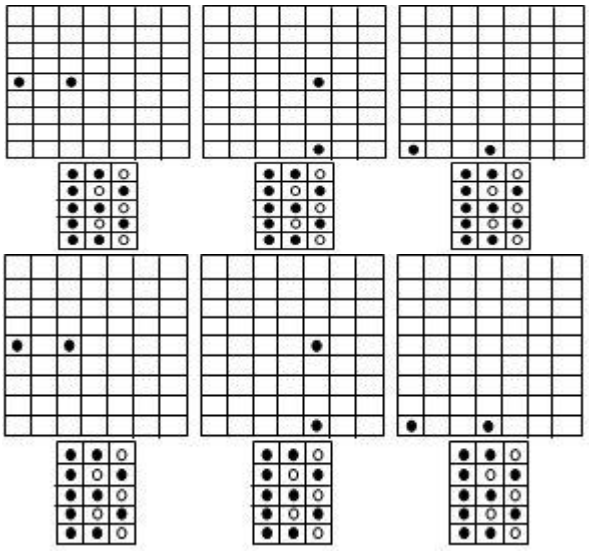

Fig. 10 Input-output associations for 3.2% partially described input pattern with second order term added

5. CONCLUSIONS

From the results that were obtained and discussed in the previous section, a conclusion can be drawn about the ability of a single layer feedforward neural network to function as a heteroassociative memory. In spite of the fact that the single layer feedforward neural network has a limited ability as compared to the multilayer feedforward neural network, if the single layer network is trained with an algorithm that has a suitable modification to the standard training algorithms (such as the suggested algorithm in this paper), it can enhance its performance as an associative memory so that it gains an ability to recall the desired stored patterns when partially described noisy inputs patterns are presented.

6. REFERENCES

[1] Zurada,J. M. 1992. Introduction to artificial neural systems, West publishing company.

[2] Sarangapani, J. 2006. Neural network control of nonlinear discrete –time systems, Taylor and Francis.

[3] Fausett, L. 1994. Fundamental of neural networks, Prentice Hall.

[4] Sudo,, A., Sato, A., and Hasegawa, O. 2009. Associative memory for online learning in noisy environments using self-organizing incremental neural network. IEEE transactions on neural networks, Vol. 20, No. 6, pp. 972, June 2009.

[5] Badri, L. 2010. Development of Neural Networks for Noise Reduction. The International Arab Journal of Information Technology, Vol. 7, No. 3, pp. 289-294, July 2010.

[6] Singh,, Y. P., Yadav, V.S., Gupta, A. and Khare, A. 2009. Bidirectional associative memory neural network method in the character recognition. Journal of Theoretical and Applied Information Technology, Vol 5, No. 4, 2009.

[7] Boutalis, Y. S. 2011. A new method for constructing kernel vectors in morphological associative memories of binary patterns. Computer Science and Information Systems, Vol. 142 8, No. 1, pp. 141-166, January 2011.

[8] Inohira,, Ogawa, E. T. and Yokoi, H. 2008. Associative Memory with Pattern Analysis and Synthesis by a Bottleneck Neural Network. Biomedical Soft Computing and Human Sciences, Vol.13, No.2, pp.27-34, 2008.

[9] Rodríguez, D., Casermeiro, E., and Lobato, J. 2007. Hopfield Network as Associative Memory with Multiple Reference Points. World Academy of Science, Engineering and Technology, Issue 0007, pp. 622-627, July 2007.

[10] Yaakobi, E. and Bruck, J. 2012. On the Uncertainty of Information Retrieval in Associative Memories. IEEE International Symposium on Information Theory Proceedings, 2012.

[11] Sommer, F. T. and Dayan, P. 1998. Bayesian Retrieval in Associative Memories with Storage Errors. IEEE transactions on neural networks, Vol. 9, No. 4, July 1998.

[12] Zeng, X. and Martinez, T. 2003. A Noise Filtering Method Using Neural Networks. International Workshop on Soft Computing Techniques in Instrumentation, Measurement and Related Applications Provo. Utah, USA, 17 May 2003.

[13] Van Gorp, J., Schoukens J., and Pintelon, R. 1998. Adding Input Noise to Increase the Generalization of Neural Networks is a Bad Idea. Intelligent Engineering Systems Through Artificial Neural Networks, Volume 8., pp. 127 – 132, 1998.

[14] Steege, F., Stephan, V., and Grob, H. 2012. Effects of Noise-Reduction on Neural Function Approximation. Proc. 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, pp. 73-78, 2012.

[15] Sussner , P. and Valle, M. 2006. Gray-scale Morphological Associative Memories. IEEE transactions on neural networks, vol. 17, no. 3, May 2006.