Bipolar Pattern Association Using A Recurrent Winner Take All Network

John E. McInroy
Dept. of Electrical Engr.
University of Wyoming
Laramie, WY 82071
mcinroy@uwyo.edu

Bogdan M. Wilamowski
Dept. of Electrical Engr.
University of Wyoming
Laramie, WY 82071
wilam@uwyo.edu

Abstract

A neural network for heteroassociative (or autoassociative) pattern recognition of input bipolar binary vectors is proposed. By combining the advantages of feedforward and recurrent techniques for heteroassociation, a simple network with guaranteed error correction is found. The heart of the network is based on a new, recurrent method of performing the Winner Take All function. The analysis of this network leads to design rules which guarantee its performance. The network is tested on a character recognition problem utilizing the entire IBM CGA character set.

1 Introduction

This paper presents a recurrent neural network for implementing the Winner Take All (WTA) function. By incorporating this network into a previously proposed neural network for bipolar pattern association [1], error correcting pattern associations can be performed easily and quickly. The resulting heteroassociative memory combines the recurrent concepts of the Hopfield network with feedforward strategies for bipolar heteroassociation. The final network is fast, requires no supervision, and is simple to implement even for a large number of stored patterns.

The heteroassociate problem treated is similar to that examined by Hao, Tan, and Vandewalle [1]. In fact, many of their elegant and useful techniques are incorporated into the design. To summarize, the problem is the following: Let an arbitrary set of well-defined associations be given by $(u_i \rightarrow z_i)$, $i = 1, ..., m$, where u_i is a bipolar vector of dimension k, and z_i is a vector (not necessarily bipolar) of dimension l. Since the input vectors are bipolar, and therefore have entries

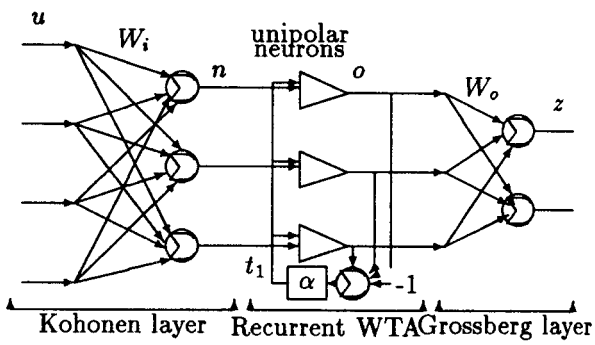


Figure 1. The neural network topology.

consisting only of ± 1, these are binary associations. For many applications, the input vector may be corrupted with noise. In the worst case, this may produce bit errors, although the proposed network will tolerate both analog and digital (bit error) noise sources. This paper will develop methods of designing the network such that the output vector will always be that association which corresponds to the minimum Hamming distance from the input vector. This error correcting ability, in particular, is extended greatly beyond that possible with Hao, Tan, and Vandewalle's network.

2 The Network Topology

The network, which consists of three parts, is depicted in Figure 1. The first layer, which is motivated by the concepts originally proposed by Kohonen [2], performs the task of roughly identifying which stored input pattern is closest (in the binary, Hamming distance sense) to the k-dimensional input vector, u. This is accomplished by configuring the input weights (W_i) so that, if the i^{th} stored pattern is the closest to the input, then the i^{th} element of the m-dimensional vector n is the maximum element of n. Since both the input

0-7803-4122-8/97 $10.00©1997 IEEE

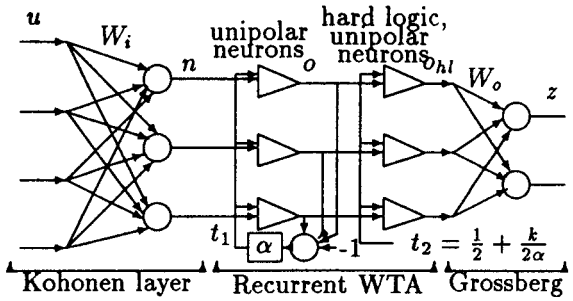


Figure 2. The neural network topology, with hard logic neurons added to eliminate possible noise.

and the stored patterns are binary, the inner product $u_i^T u$ equals $k - 2HD_{ui}$, where HD_{ui} is the Hamming distance (or number of bit errors) between the input and the i^{th} stored input pattern. Consequently, if the j^{th} stored pattern is closest (in the Hamming distance sense) to the input, then the the inner product $u_i^T u$ will be maximized when $i = j$. For this reason, if n is a vector of containing the inner products between the input and all stored input patterns, i.e.

$$n = [u_1^T u \; u_2^T u \; \ldots u_m^T u]^T, \tag{1}$$

then the i^{th} element of n will be maximum when the i^{th} stored input pattern is the closest to the input pattern. The relationship (1) can be realized by forming the weighting matrix

$$W_i = [u_1^T \; u_2^T \; \ldots u_m^T]^T, \tag{2}$$

Note that this has two extremely useful properties: (1) since the patterns are bipolar, so is W_i, thus storage of the weighting matrix is simpler than in the case of Hopfield networks; and (2) the weighting matrix does not need to be calculated, as it follows directly from the stored input patterns.

To facilitate the description of the second layer's operation, an easier to understand version will first be considered. This network (Figure 2) contains an extra layer of unipolar, hard logic neurons. They are used strictly to convert the vector o to a purely binary output vector, o_{hl}. By using a large gain, α, and a steep slope on unipolar neuron's activation function, the "winner" can be made very close to one and the "losers" can be made very close to zero. Consequently, the hard logic neurons will not be necessary for many applications. However, the network will be analyzed using the configuration shown in Figure 2 to avoid asymptotic arguments which may obscure the main points of the paper.

The WTA function inputs a vector, and then outputs which element of that vector is maximum. In a neural network, the output neuron corresponding to the maximum element of the input vector fires. In other words, if a vector $n \in \Re^{m \times 1}$ is presented to the network, then the unipolar, hard logic output vector of the network, $o_{hl} \in \Re^{m \times 1}$, will contain a single one which corresponds to the maximum element of n. All the other elements of o_{hl} will be zero. The nonlinear function, $f_{hl}(\cdot)$, which implements the hard logic neurons has the following characteristic:

$$f_{hl}(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases} \tag{3}$$

If the input to the i^{th} neuron is $x = n_i - t_2$, where n_i is the i^{th} element of n, and t_2 is a threshold, then the i^{th} hard logic neuron's output, o_{hl_i} is

$$o_{hl_i} = \begin{cases} 0, n_i < t_2 \\ 1, n_i \geq t_2 \end{cases} \tag{4}$$

First consider the case where the first layer of unipolar neurons is not present, but connections are made directly from n to the hard logic neurons. Then the WTA function can be implemented by increasing the neuron threshold, t_2, until only a single hard logic neuron fires. While this is effective, it requires "supervision" in the sense that some mechanism must be used to increase the threshold, while the output vector (o_{hl}) is monitored. When o_{hl} has only a single one, then t_2 is held constant. If a new input pattern is presented, then t_2 must be reset to a low value, and then the process is repeated. In essence, a search procedure which finds the value of t_2 corresponding to a single neuron fired must be implemented.

Rather than using a supervised search procedure, the same function will be performed by adding the first layer of unipolar neurons, along with the feedback (shown in Figure 2) which modifies the first layer's threshold, t_1. By using the results from the next section, the recurrent WTA network will automatically perform the search in a highly parallelized manner which is simple to implement with circuit technology. Consequently, the recurrent WTA network can be viewed as a fast and efficient method of finding the maximum of an m-dimensional vector. If m increases, the the only modification required is the addition of another unipolar neuron. Since the network grows so slowly with increases in m, this means that it is practical for performing parallelized, large scale searches–a functionality which is important for pattern association and many other applications.

The final layer is termed a Grossberg layer due to its similarity to the networks typically trained using Grossberg's outstar learning rule [3]. The input to the

layer, o_{hl}, will be an m-dimensional vector consisting of all zeros, except for a single element equal to 1. That element will be the ith element if the ith stored input pattern is the closest match to the network input, u. As a result, the l-dimensional network output vector, z, will equal the ith column of W_o. If

$$W_o = [z_1 \; z_2 \cdots z_m] \tag{5}$$

then the correct pattern association (i.e. $u_i \rightarrow z_i$) will be performed. Again, note that the output weights, W_o, can be analog, bipolar binary, unipolar binary, etc. Consequently, considerable freedom is available to choose a format suited to a particular implementation. Furthermore, W_o is derived directly from the output stored patterns by using (5). Finally, note that if error correcting autoassociation is desired, then $W_o = W_i^T$. This can be used to further reduce the weight storage requirements, when an autoassociative network is desired.

3 Design of the Recurrent WTA Network

The previous section delineated a method whereby error correcting bipolar heteroassociation can be elegantly performed. Most of this network was originally developed in [1], with the notable exception of the recurrent WTA network. Since [1] does not use recurrence in its WTA network, its error correcting ability is diminished. This section will show that, by correctly designing the recurrent WTA network, the correct association will always be made.

Only the recurrent WTA portion of the network, will be considered in this section since it alone remains to be designed. Due to space limitations, only brief sketchs of the proofs are presented. The proofs appear in [4].

Theorem 1 *The hard logic neuron output vector, o_{hl}, will have exactly one neuron on if the following conditions hold:*

1. *The maximum element of the vector n is at least ϵ greater than the next highest response.*
2. *The unipolar neuron activation function, $f_1(x)$ has a slope such that, if for some $x_1 < 2$, $f_1(x_1) = \frac{1}{2}(1 + k/\alpha)$ and $x_2 = x_1 - \epsilon$, then $f_1(x_2) \leq \epsilon_1$, where*

$$\epsilon_1 \leq \frac{1}{2(m-1)}[1 - k/\alpha]$$

An appropriate activation function is depicted in Figure 3.

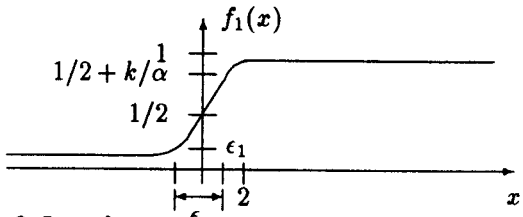


Figure 3. In order to guarantee performance, the unipolar neuron activation function must have the form shown above.

3. $f_1(0) = 1/2$.
4. *Any input pattern, u, has at least one stored pattern match with more than one-half of the bits matching.*
5. *The threshold of the hard logic neurons, t_2, equals $t_2 = \frac{1}{2}(1 + k/\alpha)$ where $\alpha > 0$.*

Corollary 2 *The recurrent WTA network will implement the WTA function if it is designed to satisfy the conditions imposed by Theorem 1.*

Discussion:

The preceding theorems provide rules which will ensure that the recurrent WTA network will perform properly. Although these design rules impose several conditions, the conditions are rather mild and can easily be implemented. For example, most of the conditions are concerned with the design of the unipolar layer's activation function. In short, these rules guarantee that its slope in the transition region is sufficiently steep to preclude the possibility that multiple neurons will be activated.

The feedback gain, α, depends greatly on the manner in which the network is implemented. A larger value of α will cause a faster rate of convergence, so ideally α should be large. Often, α may be determined by the properties of the neurons themselves.

The minimum separation between responses, ϵ, is $\epsilon = 2$ if no input vector (u) is expected to match more than one of the stored input patterns equally. However, since u is often noisy, the only way guarantee this is to put constraints on the allowable noise levels and/or the allowable stored input patterns. While these kinds of constraints can be employed with this neural network, another approach can also be used when the stored input patterns are close together and/or there are high levels of input noise. If more than one stored input pattern matches the input (u) with an identical number of matching bits, then it is impossible to tell which of the identical matches to output. If noise is present on

the vector n, then one of the matching patterns can be randomly selected by choosing a value of ϵ which is less than the separation between elements of n caused by that noise. In this case, ϵ will be much smaller than 2.

The hard logic layer of unipolar neurons is completely determined, as the threshold is the only design parameter, and it is given by the formula $t_2 = \frac{1}{2}(1 + k/\alpha)$.

4 Experimental Results

In order to test the neural network, it has been applied to a simple character recognition problem. IBM's CGA graphic representation of the ASCII character set consists of 256 characters. Each of these characters is represented in an 8 by 7 pixel array. Consequently, a total of $k = 56$ bits are used to represent each character's graphic pattern. All $m = 256$ members of the CGA ASCII set are stored in the matrix W_i, so W_i is a 256×56 bipolar binary matrix. Three output matrices, W_o, may be useful. First, a W_o which outputs an error corrected version of the input image can be found by letting $W_o = W_i^T$. In this case, the network is used for autoassociation. Note that, since W_o contains the same information as W_i, some implementations of the network may be able to use the same memory locations to store both W_o and W_i. Second, a single analog output ($l = 1$) can represent the 256 levels in a highly concise, analog form.

During the simulation, a random character from the ASCII set is chosen. Next, many of its 56 bits are randomly determined to introduce noise. A large feedback gain ($\alpha = 1000$) has been used. Uniformly distributed noise between ± 0.5 has been added to the vector n in order to separate identical responses. Since Theorem 1 gives constraints on the minimum slope needed for $f_1(\cdot)$, an activation function with a sharper slope has been used, with $gain = 30$: $f_1(x) = \frac{1}{1+e^{-gain \times x}}$

4.1 Autoassociative Experiments

Figure 4 demonstrates some of the results found for autoassociation. In each Figure, the ASCII character originally chosen is depicted in the first column. The second column depicts that character after the random bits are added to it. This noisy image forms the input, u, to the neural network. The third column shows the neural network's output after convergence. In all cases, the network converges to the ASCII character with the smallest Hamming distance from the noisy input character.

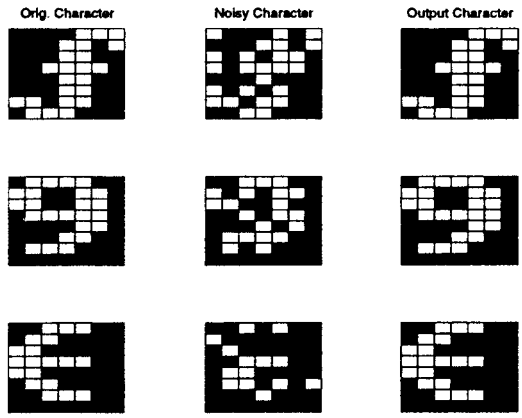


Figure 4. Three ASCII characters recognized by the neural network. The top row has 11 bit errors. The middle row also has 11 bit errors, but the next best match has only one more bit error. The bottom row contains 13 bit errors.

5 Conclusion

A new, recurrent neural network has been proposed which allows for very simple storage of patterns, and the memory capacity is large. The input weights have only binary values, in contrast to many traditional associative memories, where integer values are required. Consequently, the network is simple and cost effective to implement.

By designing the recurrent WTA portion of the network according to the rules developed herein, it will be guaranteed to produce an output which is closest (in the Hamming distance sense) to the input pattern.

References

[1] J. Hao, S. Tan, and J. Vandewalle, "Bipolar pattern association using a two-layer feedforward neural network," *IEEE Trans. On Circuit and Systems-I*, vol. CAS1-40, pp. 943–946, December 1993.

[2] T. Kohonen, "The neural phonetic typewriter," *IEEE Computer*, vol. 27, no. 3, pp. 11–22, 1988.

[3] S. Grossberg, *Studies of Mind and Brain: Neural Principles of Learning Perception, Development, Cognition, and Motor Control.* Boston: Reidell Press, 1982.

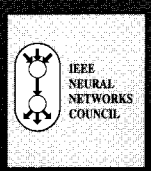
[4] J. E. McInroy and B. M. Wilamowski, "Bipolar pattern association using a recurrent winner take all network," *Submitted to Neural Processing Letters*, 1997.

1997 International Conference on Neural Networks (ICNN '97)

Westin Galleria Hotel, Houston, Texas, USA

TUTORIALS: June 8, 1997

CONFERENCE: June 9-12, 1997



Proceedings

Volume 2 of 4