Neural Pre-coding Increases the Pattern Retrieval Capacity of Hopfield and Bidirectional Associative Memories

Amir Hesam Salavati*, K. Raj Kumar*, Amin Shokrollahi* and Wulfram Gerstner†
†Brain Mind Institute *Laboratoire d'algorithmique
Ecole Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland
E-mail: {hesam.salavati,raj.kumar,amin.shokrollahi,wulfram.gerstner}@epfl.ch

***Abstract*—We consider the problem of neural association, which deals with the retrieval of a previously memorized pattern from its noisy version. The performance of various neural networks developed for this task may be judged in terms of their pattern retrieval capacities (the number of patterns that can be stored), and their error-correction (noise tolerance) capabilities. While significant progress has been made, most prior works in this area show poor performance with regard to pattern retrieval capacity and/or error correction.**

In this paper, we propose two new methods to significantly increase the pattern retrieval capacity of the Hopfield and Bidirectional Associative Memories (BAM). The main idea is to store patterns drawn from a family of low correlation sequences, similar to those used in Code Division Multiple Access (CDMA) communications, instead of storing purely random patterns as in prior works. These low correlation patterns can be obtained from random sequences by pre-coding the original sequences via simple operations that both real and artificial neurons are capable of accomplishing.

I. Introduction

In our daily routine, we encounter many cases where we have to deal with noisy information: for example, reading a document with misspelled words or trying to identify a person in a blurry picture. In both cases, the human brain does a fairly good job in retrieving the correct information from the noisy input. In the parlance of neuroscience, the brain *memorizes* an *association* between patterns (for example between the spelling of a word and its meaning, or between a face and who it belongs to), and later on we would like it to *recall* the association that a noisy input represents. An important characteristic of a neural association system is the maximum number of patterns it can memorize while still being able to tolerate a fair amount of noise during the recall process.

Two particular neural networks that have been extensively investigated in the literature are the Hopfield [7] and Bidirectional Associative Memory (BAM) [9] networks. Previous works on these networks have considered the process of memorizing a set of *random binary vectors*. The storage efficiency of these schemes are quite poor, as will be made more precise later on.

In this paper, we propose ways in which we can significantly improve the pattern retrieval capacity of such neural systems, by storing not random patterns, but structured patterns that have good "distance/correlation" properties. Our claim is that this assumption of working with structured patterns is biologically meaningful as well, since sensory inputs to the brain typically undergo several stages of neural pre-processing before being actually "stored" in the brain.

Before presenting our solutions, we briefly introduce the neural model and discuss some related works in Section II. Sections III and IV then present our two proposed methods to increase the pattern retrieval capacity of the Hopfield and BAM networks.

II. Problem Formulation

A. Neural Model

The neural network model that we consider can be represented by an undirected complete graph of N binary nodes with weighted links between each pair of nodes. Each node of the graph represents a neuron. At every instant of the process the state of node i, denoted by s_i, indicates whether or not neuron i has fired at that instant (we will not show the dependency of s_i on time to alleviate notation). In particular, we set $s_i = 1$ when the neuron fires, and $s_i = -1$ when it remains silent. The weight w_{ij} between nodes i and j denotes the binding strength (interaction) between these two neurons and can assume any real number.

At any given instance of time, a node in the network decides its state based on the inputs from its neighbors. More specifically, neuron i fires if the weighted sum

$$h_i = \sum_{j=1}^{N} w_{ij} s_j \tag{1}$$

over its input links w_{ij} exceeds a firing threshold Θ,

$$s_i = \begin{cases} 1, & \text{if } h_i \geq \Theta \\ -1, & \text{otherwise} \end{cases} . \tag{2}$$

The main task of neural association is to choose the graph weights w_{ij} such that the network is able to memorize M binary patterns of length N. In this paper, we are mostly interested in auto-association, i.e. retrieving a memorized pattern from its noisy version. In the sequel, we denote these patterns by $\{x^1, x^2, \ldots, x^M\}$, where $x^\mu = (x_1^\mu \cdots x_N^\mu)$ is a binary pattern of length N with $x_j^\mu \in \{-1, +1\}$ for all j.

The neural association consists of two phases: learning (memorizing) and recalling. In the learning phase, the network *learns* the weights based on the binary patterns that it is supposed to memorize. In the recall phase, the weights learnt remain fixed and the network is initialized with a noisy version of one of the memorized patterns [1]. Neurons then update their states according to equation (2), and the network evolves until a stable state is reached. We will primarily work with the case where the neurons are updated synchronously (i.e., all neurons are updated simultaneously at each time step); our analysis may also be extended to the case of asynchronous updates (one neuron at a time).

During the recall phase, there are two main requirements that need to be met to ensure successful operation: stability of the memorized patterns and noise tolerance. The first requirement simply means that in absence of noise, memorized patterns must be stable states of the network. More precisely, if we initialize the network with one of the memorized patterns x^γ, i.e. $s_j = x_j^\gamma$ for all $j = 1, \ldots, N$, after one time step of network evolution we would like to have $s_i = x_i^\gamma$ for all $i = 1, \ldots, N$, where each s_i is given by equation (2). The second condition requires the mechanism to be able to tolerate a fair amount of noise and settle down to the correct pattern even if initialized with a corrupted version of the pattern, i.e., if a few of the initial bits are flipped because of noise.

B. Related Works

In what is probably the most well-known work on the neural associative memory, Hopfield [7] used weights w_{ij} based on a Hebbian learning rule [2], given by

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{M} x_i^\mu x_j^\mu. \tag{3}$$

The Hopfield rule is local, in the sense that the weight between neurons i and j only depends on the i^{th} and j^{th} bit of the patterns that we would like to memorize. It is also memoryless, meaning that if a new pattern is introduced during the learning phase, the new weight matrix W only depends on the previous weight matrix and the new pattern, and not on the values of all previous patterns that were stored. These two rules translate into low computational complexity in the learning phase.

Amit et al. [1] showed that as long as $M \leq 0.138N$, the proposed model is able to memorize up to M *random* binary vectors of length N with small bit error probability in the recall phase. Here, random means that the patterns can be any of the 2^N binary vectors of length N with equal probability. Later, McEliece et al. [8] showed that if we require all patterns to be recalled with vanishing codeword error probability (as opposed to bit error probability), then the capacity is only proportional to $N/\log(N)$.

In [10], the authors used for the weights the pseudo-inverse rule [6] and showed that one can memorize up to N patterns so that they are fixed points of the dynamics, but only $N/2$ *random patterns* can be stored if one requires at least *one bit* of error correction. While this is a major improvement to the $N/\log(N)$ scaling of the pattern retrieval capacity in [8], it, however, entails significant additional computational costs because of the pseudo-inverse operation in the proposed weighting rules. Furthermore, the update rule is not local, and except for a special case, it is not memoryless either.

The pattern retrieval capacity of Hopfield networks can be improved significantly if one considers low-activity patterns, i.e. patterns where most of the neurons are silent in any given time instance. It can be shown that as the number of active neurons decreases, the number of patterns that can be stored increases [6]. Nevertheless, when required to correct a fair amount of erroneous bits, the information retrieval is not better than that of networks with balanced activity patterns.

When comparing the linear (in N) pattern retrieval capacity of neural networks with those of structurally similar codes on graphs (such as LDPC or fountain codes) from coding theory, we observe a huge gap; these codes on graphs are able to *memorize* an exponential number of patterns in terms of N. Various factors (including the simplified neural rule for "decoding") may cause this gap, but among those could be the assumption that the network should be able to memorize M *random* patterns. This assumption is usually justified by our lack of knowledge about how exactly the brain deals with input information before storing it.

However, it is known that information about the outside world undergoes several pre-processing stages before being memorized. One good example is the human visual system in which the image on the retina is first processed by the Lateral Geniculate Nucleus (LGN) of the thalamus and then by the primary visual cortex (V1) [2]. After several levels of similar processing, it reaches the point to be associated with other objects (in the learning phase) or used to recall a previous association. Therefore, it appears highly unlikely that neural patterns are completely random at the point where they are stored in the brain. Utilizing non-random patterns has the advantage that we can consider patterns that have desirable properties such as a larger minimum distance, or lower correlation, when compared to purely random patterns. Therefore, using non-random patterns might result in larger pattern retrieval capacities. We will focus in this paper on storing N pre-coded patterns in a network of N neurons.

It should be also mentioned that one can trivially memorize up to N orthogonal patterns and make sure they are fixed points of the neural association process. However, networks with orthogonal patterns are unable to correct any errors during the recall phase [6]. As a result, such methods do not satisfy the noise tolerance condition and can not be used as neural associative memory.

Recently, Berrou and Gripon [4] have also demonstrated increases in the pattern retrieval capacity of Hopfield networks, by using Walsh-Hadamard sequences used in CDMA systems to combat the noise in the network. However, there is a major drawback with their approach, since they consider a separate

[1]Here, initialization means that each bit of the noisy pattern is assigned to its corresponding node, i.e., the first bit is assigned to node 1, the second bit to node 2, and so on.

soft Maximum Likelihood (ML) decoder to deal with noise in the input. Having an extra ML decoding stage is undesirable in the setting of neural networks, in the interest of minimizing the complexity.

In contrast, our proposed approaches employ local and memoryless low-complexity learning rules, do not need any extra decoding stages during the recall phase, and are able to guarantee a reasonable amount of error correction.

C. Dealing with Interference

Consider the case where we use the Hopfield weighting rule (3). To obtain better insight on the stability condition, consider the input h_i to neuron i obtained by using (3) in (1), when we initialize the network with pattern x^γ:

$$\begin{aligned} h_i &= \sum_{j=1}^{N} w_{ij} x_j^\gamma = \frac{1}{N} \sum_{\mu=1}^{M} x_i^\mu \sum_{j=1}^{N} x_j^\mu x_j^\gamma \\ &= x_i^\gamma + \frac{1}{N} \sum_{\substack{\mu=1 \\ \mu \neq \gamma}}^{M} x_i^\mu \langle x^\mu, x^\gamma \rangle = x_i^\gamma + I_i^\gamma, \end{aligned} \tag{4}$$

where $\langle x^\mu, x^\gamma \rangle$ is the inner product of patterns x^γ and x^μ. In the above equation, the first term, i.e., x_i^γ, is the "desired term" because we would like the thresholded version of h_i (according to (2)) to be equal to the x_i^γ (since our objective is to recall pattern x^γ). The second term I_i^γ is the interference term, which depends on the correlation between pattern γ and all other patterns. Ideally, we would like the interference term be as small as possible such that it allows for recovery of x^γ.

It can be shown (details omitted for brevity) that the traditional CDMA approach of employing any family of Welch bound achieving low-correlation sequences [2] is insufficient for our purpose; smarter cancellation is needed among the summands of the second term in (4). In what follows, we will show that such intelligent cancellation occurs when we suitably pick the x^μ from the family of Gold sequences [3]. We first provide a quick overview of this sequence family.

D. Gold Sequences

In this subsection, we assume that the reader is familiar with the basics of finite fields. Let $q > 0$ be an odd integer, and $d = 2^l + 1$ for some l such that l and q are relatively prime. Also, let α be a primitive element of $\mathbb{F}_{2^q}$, and $T(\cdot)$ denote the finite field trace function from $\mathbb{F}_{2^q}$ to $\mathbb{F}_2$. We set $N = 2^q - 1$. The family of cyclically distinct Gold sequences is defined to be the set $\mathcal{G} = \{g^a | a \in \mathbb{F}_{2^q}\}$, where each $g^a = (g_1^a \cdots g_N^a)$ is a sequence of length N with

$$g_i^a \triangleq (-1)^{T(a\alpha^i) + T(\alpha^{di})}. \tag{5}$$

Hence the number of sequences in $\mathcal{G}$ is $N + 1$. For later use, we define the notation $g^a(k)$ to represent a cyclic shift of g^a by k positions (for simplicity, we will refer to $g^a(0)$ simply as g^a). Also, it can be easily verified that the Gold sequences are periodic with period N [3]. In the sequel, we will choose our patterns $\{x^1, \ldots, x^M\}$ to be either Gold sequences, or cyclic shifts of Gold sequences, depending on the value of M. The particular sequences chosen will be specified later. We set the first pattern $x^1 = g^0$, where 0 denotes the zero element in $\mathbb{F}_{2^q}$.

Before we proceed, it must be noted that generating low correlation sequences using neural networks is easily accomplished, since one can generate the Gold family using only shift registers and logical AND operations [3] (the logical AND operation may be implemented using a simple two-layer neural network, see [6]). Owing to lack of space, we will not go into much details in this regard and point the reader to [3], [5] for further details regarding how Gold sequences are generated using shift registers.

III. Hopfield Networks with Scaled Weights

In this section, we elaborate on our first solution to the problem of increasing the pattern retrieval capacity of neural networks. The proposed solution is based on the auto-associative Hopfield network with a slightly modified weighting rule. We show that using this technique, the neural network will be able to memorize $M = N$ patterns with good error correction capabilities, which is a significant improvement over [7], [8].

Our goal is to memorize the first M patterns of the Gold family with parameter q, i.e., we set $x^\mu = g^\mu$ for all $\mu = 1, \ldots, M$. Recall that $N = 2^q - 1$. The neural update rule is fixed and given by (2). Finally, we assume that the neural firing threshold Θ is equal to zero in this section.

The main idea in the proposed method is to consider a generalized learning rule,

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{M} \lambda_\mu x_i^\mu x_j^\mu, \tag{6}$$

where the scaling factors λ_μ are to be picked such that the interference term in (4) becomes small. Before we go ahead and present our choice of the λ_μ, we first examine the case of $\lambda_\mu = 1, \ \forall \mu = 1, \ldots, M$, which reduces (6) to the original Hopfield weighting rule in (3). We will show that this scheme has very good performance in terms of stability but that the error correction capability is very poor.

A. Stability Analysis with $\lambda_\mu = 1 \ \forall \ \mu$

To assess the stability property of Gold sequences, we pick the $M = N + 1$ sequences from $\mathcal{G}$ to constitute the patterns that we would like to memorize (these are all cyclically distinct, by construction). Without loss of generality, let's assume that the pattern x^γ, which we would like to recall at the moment, corresponds to the Gold pattern with $a = 0$ (see (5)). Specializing the second term of (4) for the case of

[2] In this paper, we use the terms pattern and sequence to convey the same meaning.

our patterns being Gold sequences, we obtain

$$\begin{aligned} I_i^\gamma &= \frac{1}{N}\sum_{a\in\mathbb{F}_{2^q},\ a\neq 0}(-1)^{T(a\alpha^i)+T(\alpha^{di})}\sum_{j=1}^{N}(-1)^{T(a\alpha^j)} \\ &= \frac{(-1)^{T(\alpha^{di})}}{N}\left(M-1+\sum_{j=1,\neq i}^{N}\sum_{\substack{a\in\mathbb{F}_{2^q}\\ a\neq 0}}(-1)^{T(a(\alpha^i+\alpha^j))}\right) \\ &= \frac{M-N}{N}(-1)^{T(\alpha^{id})} = \frac{M-N}{N}x_i^\gamma = \frac{1}{N}x_i^\gamma. \end{aligned} \tag{7}$$

The first equality in (7) holds because $\alpha^i+\alpha^j\neq 0$ for $i\neq j$ (since α is primitive element of $F(2^q)$), and then since the trace function is a linear form:

$$\sum_{a\in\mathbb{F}_{2^q}}(-1)^{T(ab)}=0,\ \text{if } b\neq 0. \tag{8}$$

Combining (7) and (4), we see that for the Gold family with $M=N+1$ we have $h_i=\frac{M}{N}x_i^\gamma$. Therefore, the sign of h_i would be equal to that of x_i^γ, i.e., the network is stable for $M=N+1$ when Gold sequences are memorized. Furthermore, we can also show that setting M to be an integer multiple of $(N+1)$, say $M=\delta(N+1)$, $\delta\in\mathbb{Z}^+$, and picking our patterns to be $\mathcal{G}$ and $(\delta-1)$ sets of cyclic shifts of all sequences in $\mathcal{G}$ will also lead to stability. Due to lack of space, we do not go into details in this regard.

B. Performance in the Presence of Noise when $\lambda_\mu = 1\ \forall\ \mu$

To investigate the performance in the presence of noise, we first consider the simple case of a single bit error in the input to the network. Suppose that we feed in the pattern x^γ which corresponds to the Gold sequence with $a=0$ (see (5)), where position k has been flipped. In other words, our initial pattern is $s=x^\gamma+e$, where the error vector e is equal to zero in all positions except for the k^{th} element e_k, and $e_k=-2x_k^\gamma$. Substituting this input pattern into (4) and using (7), we obtain:

$$h_i = \frac{M}{N}x_i^\gamma-\frac{2}{N}\sum_{\mu=1}^{M}x_i^\mu x_k^\mu x_k^\gamma. \tag{9}$$

The term $E_k^\gamma=\sum_{\mu=1}^{M}x_i^\mu x_k^\mu x_k^\gamma$ reduces to Mx_i^γ if $k=i$. When $k\neq i$, we obtain using (8) that $E_k^\gamma=0$. From (9) and the above, we obtain

$$h_i=\begin{cases}-\frac{M}{N}x_i^\gamma, & \text{if } i=k\\ \frac{M}{N}x_i^\gamma, & \text{Otherwise}\end{cases}. \tag{10}$$

This essentially means that no error correction is possible: all flipped bits remain erroneous and all correct bits remain correct as the system evolves.

C. Choosing λ_μ to Ensure Good Error Correction

From the above, the need to choose the λ_μ intelligently becomes clear: we need to make sure that pair-wise pattern correlations cancel each other out when summing up over all patterns, in the second term of (4). Towards this end, we simply pick $\lambda_\mu=(-1)^{T(\alpha^{-\mu})}$; since this term alternates in sign for different μ's, the interference terms in the summation in (4) cancel each other out. This intuition is validated by the simulation results below, where we show that using this choice of λ_μ, we will be able to memorize $M=N$ patterns drawn from the Gold sequence family.

D. Simulation results

We consider three different Gold families, with parameters $q=5$, $q=7$ and $q=9$. These correspond to having $N=31$, $N=127$ and $N=511$ respectively. As mentioned earlier, the first $M=N$ patterns of the Gold family are selected. We consider various number of initial (uniformly random) bit flips and evaluate the pattern error rate. The results are reported in Fig. 1 and are compared to the pseudo-inverse method proposed in [10].[3] As can be seen, the number of initial bit flips that can be corrected by the proposed method depends on the network size N. With larger N's, we are able to tolerate larger amounts of noise. For instance, with $N=31$, the suggested method is able to correct one bit flip, while with $N=511$, the proposed solution can correct up to 40 bits of error. Hence, our method results in a robust associative memory with $M=N$, which is significantly better than previous works both in terms of capacity [8] and noise tolerance [10].

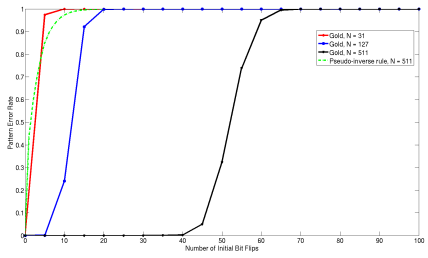

Fig. 1. Pattern error rate for our scaled Hopfield network with $M=N$, compared to the pseudo-inverse method [10] with $M=\lfloor N/2\rfloor-1$.

IV. Bidirectional Associative Memory with two Stage Recall

In this section, we consider the case where we would like to memorize M suitably chosen patterns using a neural network with $M+N$ neurons. The number of patterns stored per neuron $\frac{M}{M+N}$ will be seen to tend to 1 as M grows and N remains fixed. For the original result of Hopfield, recall that the number of patterns stored per neuron is only 0.138. Thus our result represents a seven-fold increase in capacity, in the limiting case. Further, our proposed scheme will be able to correct up to $N_{\text{err}}=\lfloor(N-\sqrt{2(N+1)}+1)/2\rfloor$ errors (bit flips).

In fact, the solution presented in this section is very similar to winner-takes-all neural networks and adaptive resonance algorithms [6]. However, the proposed method has the advantage that it does not need any lateral connections to enforce the winner-takes-all rule; because of its special structure, the same phenomenon occurs automatically. Furthermore, in contrast to

[3]For clarity purposes, we have illustrated only the pattern error rate corresponding to $N=511$ for the pseudo-inverse method [10] as the results for $N=31$ and $N=127$ were quite the same.

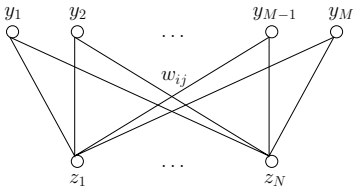

Fig. 2. A bipartite neural network

$N(N_{\text{err}})$ \ ϵ	1	2	5
31(21)	64	96	192
127(56)	256	384	768
511(240)	1024	1536	3072

TABLE I
PATTERN RETRIEVAL CAPACITY OF THE BAM, FOR SEVERAL N, ϵ. ALSO SHOWN IS THE NUMBER OF BIT-FLIP ERRORS TOLERATED N_{ERR}.

adaptive resonance algorithms, the suggested approach is not highly sensitive to input noise.

Consider the neural network shown in Fig. 2, composed of a complete bipartite graph between neurons $z_1, \ldots, z_N$ and $y_1, \ldots, y_M$. We choose the weights w_{ij} between z_i and y_j to be x_i^j, i.e., the i^{th} bit of the j^{th} pattern. Suppose that the neurons $z_1, \ldots, z_N$ are initialized with a possibly corrupted pattern X. Our recall process to recover the correct pattern after error correction comprises of two stages: a forward and a backward iteration. In the sequel, we will assume that $M = \delta(N+1)$, for some fixed positive integer $\delta < N$. In this case, our set of patterns is chosen to include $\mathcal{G}$, and $(\delta - 1)$ sets of cyclic shifts of all sequences in $\mathcal{G}$ (where the cyclic shifts used for each set may be arbitrary, as long as they are distinct).

A. Forward Iteration

Given our particular choice of $w_{ij} = x_i^j$, the input to neuron y_j is equal to the correlation between X and pattern x^j, denoted as $\langle X, x^j \rangle$. Let x^j correspond to pattern $g^a(k)$, for some $a \in \mathbb{F}_{2^q}$ and $k \geq 0, k \in \mathbb{Z}$. Without loss of generality, assume that the input pattern X corresponds to a corrupted version of x^1, i.e., $X = g^0 + e$, where e is an error vector. The input to neuron y_j is

$$\langle g^0 + e, g^a(k) \rangle = \langle g^0, g^a(k) \rangle + \langle e, g_a(k) \rangle. \qquad (11)$$

For the case that $k = 0, a = 0$, we have that $\langle g^0, g^a(k) \rangle = N$. For all other cases, it is known from the property of the Gold sequence family that $|\langle g^0, g^a(k) \rangle| \leq \sqrt{2(N+1)}+1$ [3]. Now consider the other correlation term in (11), viz., $\langle e, g_a(k) \rangle$. For simplicity, let e correspond to a single bit error in X, at position ℓ, i.e., $e = [0 \ldots 0 - 2g_\ell^0 0 \ldots 0]$. Using the periodicity property [3], we may evaluate

$$\langle e, g^a(k) \rangle \quad = \quad (-2)(-1)^{T(\alpha^{d\ell}) + T(a\alpha^{\ell+k}) + T(\alpha^{d(\ell+k)})} = \pm 2.$$

Suppose that we set the threshold of the y_i neurons to be midway between $\sqrt{2(N+1)}+1$ and N (we assume $N \geq 5$, so that $\sqrt{2(N+1)} + 1 < N$). From the above, it is clear that only neuron y_1 will fire (and all others remain silent), as long as the number of errors is less than or equal to $N_{\text{err}} = \lfloor (N - \sqrt{2(N+1)} + 1)/2 \rfloor$.

B. Backward Iteration

Following this, we do a backward iteration, where the z_i neurons now update their values based on the states of the y_j neurons. Given that only y_1 is active, we would like to recover x^1 as the states of the z_i neurons. For reasons that will become clear later, we now assume that the firing levels of the y_j neurons are $\{-1, +\epsilon\}$, for some constant $\epsilon > 0$ (as opposed to the more traditional $\{\pm 1\}$ firing levels). For ease of exposition, consider first the case of $\delta = 2$, i.e., $M = 2(N+1)$. In this case, we use the set of $(N+1)$ cyclically distinct sequences from $\mathcal{G}$, along with cyclic shifts of all sequences in $\mathcal{G}$ by 1 (say). The input to neuron z_1 is given by

$$\begin{aligned} h_1 \quad &= \quad \epsilon(-1)^{T(\alpha^{d\cdot 1})} + (-1) \sum_{a \in \mathbb{F}_{2^m}, a \neq 0} (-1)^{T(a\alpha^1 + \alpha^{d\cdot 1})} \\ &\quad + (-1) \sum_{a \in \mathbb{F}_{2^m}, a \neq 0} (-1)^{T(a\alpha^2 + \alpha^{d\cdot 2})} \\ &= \quad (\epsilon + 1)(-1)^{T(\alpha^{d\cdot 1})} + (-1)^{T(\alpha^{d\cdot 2})}. \end{aligned}$$

We hence see that for values of $\epsilon > 0$ and a firing threshold of $\Theta_z = 0$, neuron z_1 recovers the correct value of $x_1^1 = (-1)^{T(\alpha^{d\cdot 1})}$ in this case. A similar analysis for all other neurons z_j shows that the entire pattern x^1 is recovered as the states of the neurons z_j. It is clear that as we include more patterns, we will have to increase the value ϵ. However, the scaling of ϵ is quite reasonable in terms of the number of patterns: for $M = \delta(N+1)$, we need $\epsilon > \delta - 2$. The intuition behind the higher range for the firing levels is that in order to increase the pattern retrieval capacity, the neurons need to increase their "operational signal-to-noise ratio." The pattern retrieval capacity of this scheme for a few representative values of N and ϵ is shown in Table I.

REFERENCES

[1] D. Amit, H. Gutfreund, H. Sompolinsky, *Storing infinite numbers of patterns in a spin-glass model of neural networks*, Physic. Rev. Lett., Vol. 55, 1985, pp. 1530-1533.

[2] P. Dayan, L.F. Abbott, *Theoretical neuroscience: computational and mathematical modeling of neural systems*, MIT Press, 2004.

[3] R. Gold, *Optimal binary sequences for spread spectrum multiplexing*, IEEE Trans. Inf. Theory, Vol. 13, No. 4, 1967, pp. 619-621.

[4] C. Berrou, V. Gripon, *Coded Hopfield Networks*, Proc. Symp. on Turbo Codes and Iterative Information Processing, pp. 15, 2010.

[5] T. Helleseth and P. V. Kumar, *Codes and Sequences over $\mathbb{Z}_4$ - A Tutorial Overview*, in Difference Sets, Sequences and their Correlation Properties, NATO Science Series, Kluwer Academic Publishers, Dordrecht, 1999.

[6] J. Hertz, A. Krogh, R. G. Palmer, *Introduction to the theory of neural computation*, USA: Addison-Wesley, 1991.

[7] J. J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proc. NatL Acad. Sci., Vol. 79, 1982.

[8] R. McEliece, E. Posner, E. Rodemich, S. Venkatesh,*The capacity of the Hopfield associative memory*, IEEE Trans. Inf. Theory, July 1987.

[9] F.T. Sommer, G. Palm, *Improved bidirectional retrieval of sparse patterns stored by Hebbian learning*, Neural Networks, Vol. 12, 1999, pp. 281297.

[10] S. S. Venkatesh, D. Psaltis, *Linear and logarithmic capacities in associative neural networks*, IEEE Trans. Inf. Theory, Vol. 35, No. 3, 1989, pp. 558-568.