Exponential Pattern Retrieval Capacity with Non-Binary Associative Memory

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

***Abstract*—We consider the problem of neural association for a network of non-binary neurons. Here, the task is to recall a previously memorized pattern from its noisy version using a network of neurons whose states assume values from a finite number of non-negative integer levels. Prior works in this area consider storing a finite number of *purely random* patterns, and have shown that the pattern retrieval capacities (maximum number of patterns that can be memorized) scale only linearly with the number of neurons in the network.**

In our formulation of the problem, we consider storing patterns from a suitably chosen set of patterns, that are obtained by enforcing a set of simple constraints on the coordinates (such as those enforced in graph based codes). Such patterns may be generated from purely random information symbols by simple neural operations. Two simple neural update algorithms are presented, and it is shown that our proposed mechanisms result in a pattern retrieval capacity that is *exponential* in terms of the network size. Furthermore, using analytical results and simulations, we show that the suggested methods can tolerate a fair amount of errors in the input.

I. Introduction

The mid 20^{th} century saw the publication of two pioneering works in the fields of neural networks and coding theory, respectively those of McCulloch and Pitts in 1943 [2] and Shannon in 1948 [1]. In their modern avatars, graphical models and algorithms are seen to play a major role in both areas. In spite of a lot of common models and techniques, there is seen to be a clear quantitative dichotomy in terms of results in these two fields. While coding theory shows that a number of "codewords" that are exponential in terms of their length can be reliably stored, transmitted and decoded, the best pattern retrieval capacities that can be obtained in a neural network scale only linearly with the length of the patterns.

One reason for this vast difference in the achievable results is that neurons in a neural network are restricted to employ only simple operations such as linear combinations of inputs, and thresholding. This rules out using a lot of techniques that are key in terms of achieving the exponential scaling results of coding theory. The natural question that arises is whether one can increase the storage capacity of neural networks beyond the current linear scaling results. We answer this question affirmatively in this paper, by showing that in fact, an *exponential* scaling is possible.

In contrast to the traditional approach of using bipolar signaling levels for the neurons [6], we will work with the case of neurons that can assume a finite set of integer values $\mathcal{S} = \{0, 1, \ldots, S-1\}$ for their states. One way of interpreting the integer states is to think of the short-term (normalized) firing rate of a neuron as its output. We restrict the operations at each neuron to a linear summation over the inputs, and a possibly non-linear "thresholding" operation. In particular, suppose that a neuron x updates its state based on the states of its neighbors $\{s_i\}_{i=1}^n$. Neuron x first computes the weighted sum

$$h = \sum_{i=1}^{n} w_i s_i,$$

where w_i denotes the weight of the input link from s_i, and then updates its state as $x = f(h)$, where $f : \mathbb{R} \to \mathcal{S}$ is a possibly non-linear function from the field of real numbers $\mathbb{R}$ to $\mathcal{S}$.

In brief, neural associative memory aims to memorize C_{max} patterns of length n (auto-associative memory) or associate C_{max} patterns of length n with C_{max} patterns of length k (hetero-associative memory). In both cases, we are given a network of neurons (binary or non-binary) whose interconnections can be modeled by a weighted complete graph. The goal is to determine the weight matrix (termed the *learning phase*) such that the network is able to recall a large number of memorized patterns (during the *recall phase*), while being able to tolerate a fair amount of noise.

In what is probably the most well-known work in this area, Hopfield [3] used the Hebbian learning rule [9] to introduce an auto-associative neural mechanism, that is known as the Hopfield network. In the Hopfield network, patterns are binary vectors of length n, and it was shown later by Amit et al. [5] that the network is able to memorize C_{max} such *random* patterns with vanishing bit error probability as long as $C_{max} \leq 0.138n$. 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. [15] 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 $n/\log(n)$. These results were extended to the case of a sparse regular neural network in [13].

In [18], using the pseudo-inverse rule [6] to determine the weight matrix, the authors 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. Although this was a

significant improvement to the $n/\log(n)$ scaling of the pattern retrieval capacity in [15], it comes at the price of much higher computational complexity.

It is known that as the number of active neurons decreases (low activity neurons), 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.

While most previous works focus on improving the pattern retrieval capacity of neural associative memories when storing random patterns, there has recently been some efforts to employ *structured (non-random) patterns* in order to increase the pattern retrieval capacity. Berrou and Gripon [7] have demonstrated considerable improvements in the pattern retrieval capacity of Hopfield networks, by utilizing Walsh-Hadamard sequences used in CDMA systems to combat the noise in the network. However, a drawback in their work is that a separate soft Maximum Likelihood (ML) decoder is used for noise removal in the input, which is undesirable in the interest of minimizing complexity.

Following the same approach of using structured patterns, in [16] we introduced two novel mechanisms of neural association that employ binary neurons to memorize patterns with low correlation properties. The network itself is very similar to that of Hopfield, with a slightly modified weighting rule. The proposed methods employ low-complexity learning rules and do not need any extra decoding stage during the recall phase. Using computer simulations, it was shown that the pattern retrieval capacity of the proposed model is $C_{max} = n$, while being able to correct a fair number of erroneous input bits.

Extensions to non-binary associative memories have also been considered previously. Hopfield himself showed in [4] that similar to the binary case, neurons with continuous states (any value between -1 and 1) can also perform collective behaviors, such as memorizing a set of *random* patterns. However, in the context of auto-associative networks with a response between -1 and 1, the pattern retrieval capacity will be less than that of a network of binary neurons [4].

Jankowski et al. have considered multi-state complex-valued neurons as a means of implementing associative memory [20]. The authors use the traditional Hopfield network structure and weight determination mechanism with complex-valued neurons. They approximate the pattern retrieval capacity as $C_{max} = \alpha n$ with α being a constant less than 0.15. Furthermore, they show that α decreases as the number of complex levels increases. In [14], the authors use the same neural model as that of [20] but show using simulations that up to $C_{max} = n$ patterns may be stored using a different method to determine the neural weights. However the complexity of the weight computation mechanism is quite high. Another extension of the Jankowski's model to admit $C_{max} = n$ pattern retrieval capacity in [21] uses a Modified Gradient Descent learning Rule (MGDR) to determine the weights.

In this paper, we follow the same approach as in our previous paper [16], and aim to memorize *structured patterns* as opposed to purely random ones. However, instead of using binary neurons and low correlation sequences as in [16], we employ neurons with non-binary integer valued states and pick our patterns to be those that satisfy some simple constraints. More precisely, if each neuron can have a state from the set of integers $\mathcal{S} = \{0, 1, \ldots, S-1\}$, instead of considering all possible S^n such patterns, we pick a subset of them by forcing the patterns to satisfy m constraints. We suitably pick the constraints such that the number of patterns in the given subset is exponential in terms of n (i.e., is equal to ℓ^n, for some $\ell > 1$). As long as the corresponding constraint matrix satisfies some expansion properties, we show that our proposed neural network can memorize these exponential number of patterns, while being able to correct a reasonable number of input errors. Furthermore, the error correction procedure is very simple and has low complexity, which makes it an appropriate choice for being implemented in real neural networks.

II. The Storage and Recall Process

We consider the case of storing C_{max} random patterns $\mathcal{X} = \{x^1, x^2, \ldots, x^{C_{max}}\}$ in a neural network. Let each pattern be denoted by $x^i = (x_1^i, \cdots, x_n^i)$, where the values x_j^i belong to the alphabet $\mathcal{S}$. As mentioned previously, we will intelligently choose the particular patterns $\mathcal{X}$ that we wish to store, as opposed to a random subset of all possible $\mathcal{S}^n$ patterns. In the parlance of error correcting codes, this amounts to introducing "redundancy" among the patterns $\mathcal{X}$ to facilitate the detection and correction of errors (noise) in the patterns. In order to pick a "good" set of patterns, we follow the approach of low density parity check (LDPC) codes , and define $\mathcal{X}$ to comprise the set of all vectors $x \in \mathcal{S}^n$ that satisfy the system of linear equations

$$Hx = b, \tag{1}$$

where H is an $m \times n$ integer matrix with each entry either 0 or 1, and b is an $m \times 1$ integer vector. We will further choose H to be a sparse matrix, i.e., H has a small fixed number of ones, say d_c and d_p in each row and each column, respectively. The problem of determining the set of integer solutions to such systems of linear equations has been studied previously, see for example [8]. In particular, it is known that the set of solutions forms a lattice translate, and can hence be generated from purely random sequences through simple neural processing. Details regarding this mapping are the subject of future work, and are beyond the scope of this paper. We will focus solely on the problem of storing $\mathcal{X}$ in the neural network in the sequel.

Suppose that we have fixed a particular choice of H and b, that in turn fixes a particular $\mathcal{X}$. We now focus on the recall process of our neural network, when fed with a noisy input. The recall process can be described using the graph in Fig. 1. A possibly noisy version of an input pattern x^i (some $i = 1, \ldots, C_{max}$) is initialized as the states of the *pattern neurons* $x_1, \ldots, x_k$ in Fig. 1. We assume that the noise is integer valued and additive, and that the value of the noise added to the pattern is clipped to either 0 or $S-1$, when the sum of the noise and input is less than 0 or greater than $S-1$, respectively. With a slight abuse of notation, this is

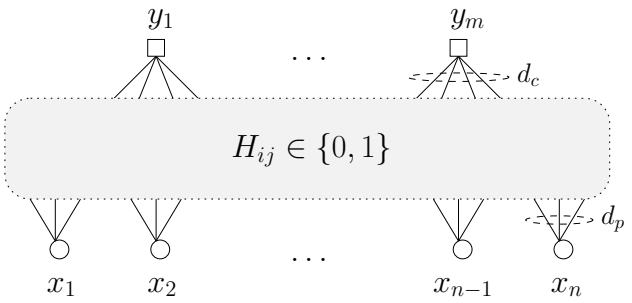

Fig. 1. Storage and recall in the neural network

represented as $x_j = x_j^m + z_j$, for some integer noise z_j. Our task in the sequel will be to "remove" this noise z_j to obtain the desired pattern x^m as the states of the pattern neurons. This task will be accomplished by exploiting the fact that we have chosen the patterns $\mathcal{X}$ to satisfy the set of constraints in (1). We adopt an iterative algorithm, implemented using the graphical structure of Fig. 1. The m *constraint neurons* $\{y_i\}_{i=1}^m$ are connected to the pattern neurons $\{x_j\}_{j=1}^n$, with H_{ij} representing the weight of the link between x_j and y_i. Hence d_c and d_p, as previously defined, denote the number of edges (degree) connected to the constraint and pattern neurons respectively. The algorithm comprises a series of forward and backward iterations, described below:

- *Forward iteration:* Each constraint neuron y_i computes a weighted sum of its inputs $h_i = \sum_{j=1}^n H_{ij}x_j$, and sets its value according to

$$y_i = \begin{cases} 1, & h_i < b_i \\ 0, & h_i = b_i \\ -1, & \text{otherwise} \end{cases}.$$

  It is clear that if the states of the pattern neurons x_i correspond to a pattern from $\mathcal{X}$ (i.e., the noise-free case), then for all $i = 1, \ldots, m$ we have $y_i = 0$ [1].
- *Backward iteration:* Each neuron x_j computes

$$g_j = \frac{\sum_{i=1}^m H_{ij}y_i}{d_p}.$$

  Roughly speaking, the quantity g_j can be interpreted as feedback to pattern neuron x_j from the constraint neurons, where the sign of g_j provides an indication of the sign of the noise that affects x_j, and $|g_j|$ indicates the confidence level in the decision regarding the sign of the noise. Once the pattern neurons compute the g_j, one can come up with many strategies regarding how they update their values. We consider the following two strategies in this paper:

  1) The pattern neurons implement a *winner-take-all* strategy, which is well known and studied among the neuroscience community [6]. Define

$$j^* = \arg\max_j |g_j|.$$

     If $g_{j^*} \neq 0$, then set $x_{j^*} = x_{j^*} + \text{sign}(g_{j^*})$. Notice that although collaboration is needed among the $\{x_j\}$ neurons to implement this decision, simple neural circuits to implement this strategy are well known [6].
  2) The pattern neurons make local decisions regarding updating their values, similar to the *bit-flipping* algorithm of [11]. We fix a threshold value $\gamma > 0$. For each $j = 1, \ldots, n$, $x_j = x_j + \text{sign}(g_j)$, if $|g_j| > \gamma$.

[1]Note that although we do not allow neurons to have negative outputs, the set of outputs $\{-1, 0, 1\}$ can be easily implemented by sending $\{0, 1, 2\}$ as the neural output and shift the response in the pattern neurons during the neural update. The whole shifting process can be translated into a modified firing threshold for each neuron which depends on its incoming degree.

III. Analysis of the Proposed Algorithm

In this section we focus on the winner-take-all version of our algorithm, and prove that it is guaranteed to correct a certain number of errors. Before turning to an analysis of the algorithm, we first show that the number of patterns that satisfy (1) and are hence stored by the suggested neural network is exponential in the length n, which is a significant improvement over the best current associative memories that only allow a linear number (in n) of patterns to be stored.

A. Number of Patterns and Distance Properties

In order to prove the subsequent results, we will need to employ *bipartite expander graphs* (as in [11], [12], [19]), which we now define.

Definition 1: A regular (d_p, d_c, n, m) bipartite graph H is a bipartite graph between n pattern nodes of degree d_p and m constraint nodes of degree d_c.

Definition 2: An $(\alpha n, \beta d_p)$-expander is a (d_p, d_c, n, m) bipartite graph such that for any subset $\mathcal{P}$ of pattern nodes with $|\mathcal{P}| < \alpha n$ we have $|\mathcal{N}(\mathcal{P})| > \beta d_p |\mathcal{P}|$ where $\mathcal{N}(\mathcal{P})$ is the set of neighbors of $\mathcal{P}$ among the constraint nodes.

The following result from [19] shows the existence of families of expander graphs with parameter values that are relevant to us.

Theorem 1: Let d_c, d_p, m, n be integers, and let $\beta < 1 - 1/d_p$. There exists a small $\alpha > 0$ such that if H is a (d_p, d_c, n, m) bipartite graph chosen uniformly at random from the ensemble of such bipartite graphs, then H is an $(\alpha n, \beta d_p)$-expander with probability $1 - o(1)$, where $o(1)$ is a term going to zero as n goes to infinity.

We now show that there exist an exponential number of patterns satisfying (1), when H is randomly chosen from the $(d_p, d_c, n, m)-$regular bipartite ensemble.

Theorem 2: Let neurons have integer-valued states between 0 and $S-1$, and each row of the $m \times n$ matrix H contain exactly d_c ones (with the remaining entries being zero). Then, there exists a column vector b such that the system of linear equations $Hx = b$ has an exponential number of solutions.

Proof: Since each component of the vector x can take any integer value from 0 to $S-1$, we have a total number of S^n such vectors. Suppose $y = Hx$. Then each component of y is between 0 and $d_c(S-1)$. Hence, there are at most $[d_c(S-1)+1]^m$ of such vectors y. Now if each component

of the vector b can take any integer value from 0 to $S'-1$, then for $S' \geq d_c S$ there exists a vector b such that the system of equations $Hx = b$ has an integer solution and $S^n/[d_c(S-1)+1]^m \geq S^n/(d_c S)^m$ vectors of length n satisfy the set of equations. Therefore, for $m = n/2$ if $S > d_c$ we will have an exponential number of solutions for the system of equations. More generally, if $\log(S) > \frac{m}{n-m}\log(d_c)$ we can find a b such that we have an exponential number of solutions. ■

Next, we present a sufficient condition such that the minimum Hamming distance[2] between these exponential number of patterns is not too small.

Theorem 3: Let H be a $(d_p, d_c, n, m)-$regular bipartite graph, that is an $(\alpha n, \beta d_p)$ expander. Let $\mathcal{X}$ be the set of patterns obtained as a solution to (1), as before. If $\beta > \frac{1}{2} + \frac{1}{4d_p}$, then the minumum distance between the patterns is at least $\lfloor \alpha n \rfloor + 1$.

Proof: The proof of the above theorem is based on expansion properties of H and is omitted due to lack of space. ■

B. Error-Correction Capability

We prove the error correction capability of the winner-take-all algorithm in two steps: first we show that in each iteration, only code neurons that are corrupted by noise will be chosen by the winner-take-all strategy to update their state. Then, we prove that the update is in the right direction, i.e. toward removing noise from the neurons.

Lemma 1: If the constraint matrix H is an $(\alpha n, \beta d_p)$ expander and the original number of erroneous neurons are less than $e_{min} = \lfloor \frac{\beta}{1-\beta} \rfloor$, then in each iteration of the winner-take-all algorithm only the corrupted pattern nodes update their value and the other nodes remain intact. For $\beta = 3/4$, the algorithm will always pick the correct node if we have two or fewer erroneous nodes.

Proof: For simplicity, we restrict our attention to the case $\beta = 3/4$. If we have only one node x_i in error, it is obvious that the corresponding node will always be the winner of the winner-take-all algorithm unless there exists another node that has the same set of neighbors as x_i. However, this is impossible as because of the expansion properties, the neighborhood of these two nodes must have at least $2\beta d_p$ members which for $\beta = 3/4$ is equal to $3d_p/2$. As a result, no two nodes can have the same neighborhood and the winner will always be the correct node.

In the case where there are two erroneous nodes, say x_i and x_j, let Q be the set $\{x_i, x_j\}$ and $N(Q)$ be the corresponding neighborhood on the constraint nodes side. Furthermore, assume x_i and x_j share $d_{p'}$ of their neighbors so that $|N(Q)| = 2d_p - d_{p'}$. First of all note that because of the expansion properties and for $\beta = 3/4$:

$$|N(Q)| = 2d_p - d_{p'} > 2\beta d_p \Rightarrow d_{p'} < d_p/2.$$

[2]Two (possibly non-binary) $n-$length vectors x and y are said to be at a Hamming distance d from each other if they are coordinate-wise equal to each other on all but d coordinates.

Now we have to show that there are no nodes other than x_i and x_j that can be the winner of the winner-take-all algorithm. To this end, note that only those nodes that are connected to $N(Q)$ will receive some feedback and can hope to be the winner of the process. So let's consider such a node x_ℓ that is connected to d_{p_ℓ} of the nodes in $N(Q)$. Let Q' be $Q \cup \{x_\ell\}$ and $N(Q')$ be the corresponding neighborhood. Because of the expansion properties we have $|N(Q')| = d_p - d_{p_\ell} + |N(Q)| > 3\beta d_p$. Thus:

$$d_{p_\ell} \quad < \quad d_p + |N(Q)| - 3\beta d_p = 3d_p(1-\beta) - d_{p'}.$$

Now, note that the nodes x_i and x_j will receive some feedback from at least $d_p - d_{p'}$ edges because those are the edges that are uniquely connected to them and noise from the other erroneous nodes cannot cancel them out. Since $d_p - d_{p'} > 3d_p(1-\beta) - d_{p'}$ for $\beta = 3/4$, we conclude that $d_p - d_{p'} > d_{p_\ell}$ which proves that no node outside Q can be picked during the winner-take-all algorithm as long as $|Q| \leq 2$ for $\beta = 3/4$. ■

In the next lemma, we show that the state of erroneous neurons is updated in the direction of reducing the noise.

Lemma 2: If the constraint matrix H is an $(\alpha n, \beta d_p)$ expander and the original number of erroneous neurons is less than $e_{min} = \lfloor \frac{\beta}{1-\beta} \rfloor$, then in each iteration of the winner-take-all algorithm the winner is updated toward reducing the noise.

Proof: As before, we only focus on the case $\beta = 3/4$. When there is only one erroneous node, it is obvious that all its neighbors agree on the direction of update and the node reduces the amount of noise by one unit.

If there are two nodes x_i and x_j in error, since the number of their shared neighbors is less than $d_p/2$ (as we proved in the last lemma), more than half of their neighbors agree on the direction of update. Therefore, whoever the winner is will be updated to reduce the amount of noise by one unit. ■

From Lemmas 1 and 2, it is clear that if H is an $(\alpha n, \beta d_p)$ expander, then the winner-take-all algorithm is guaranteed to correct at least $e_{min} = \lfloor \frac{\beta}{1-\beta} \rfloor$ positions in error, irrespective of the magnitudes of the errors. Similar statements can be made for the bit-flipping algorithm as well. However, in that case there will be a small probability of error if the number of erroneous nodes is larger than 1 because of the chance that noise terms cancel each other out. We have omitted the proof due to lack of space. Compared to the winner-take-all algorithm, the bit-flipping method has less complexity but a worser performance as well.

C. Choice of Parameters

In order to put together the results of the previous two subsections and obtain a neural associative scheme that stores an exponential number of patterns and is capable of error correction, we need to carefully choose the various relevant parameters. We summarize some design principles below.

- From Theorems 1 and 3, the choice of β depends on d_p, according to $\frac{1}{2} + \frac{1}{4d_p} < \beta < 1 - \frac{1}{d_p}$.
- Choose $d_c, S, S', r \triangleq m/n$ such that $S^n > (d_c S)^{rn}$ and $S' > d_c S$, so that Theorem 2 yields an exponential number of patterns.

- For a fixed α, n has to be chosen large enough so that an $(\alpha n, \beta d_p)$ expander exists according to Theorem 1, and so that $\alpha n/2 > e_{min} = \lfloor \frac{\beta}{1-\beta} \rfloor$.

Once we choose a judicious set of parameters according to the above requirements, we have a neural associative memory that is guaranteed to recall an exponential number of patterns even if the input is corrupted by errors in two coordinates. Our simulation results will reveal that a greater number of errors can be corrected in practice.

IV. Simulation Results

We have assessed the performance of the proposed methods via simulations. Notice that when the network in Fig. 1 is initialized with a noisy version of the i^{th} pattern $x = x^i + z$, the check neurons compute $Hx - b = Hz$. Hence the states of the check neurons depend only on the noise, or to be more precise, on sign(Hz). Therefore, instead of first solving the systems of linear equations $Hx = b$, corrupting the solution with noise and then get rid of $Hx - b$ in the first round of error correction, we can evaluate the performance of the error correction method by picking an expander matrix H, initializing the code neurons with just noise and setting b to be the all zero vector. We check whether the algorithm will be able to get rid of noise in a few iterations. Otherwise, an error is declared.

Fig. 2 illustrates the performance of the winner-take-all and bit flipping algorithm as a function of initial number of erroneous bits. Simulation parameters are $m = 300$, $n = 600$, $d_p = 5$, $d_c = 10$, $\beta = 3/4$ and $\gamma = 0.8$ (the update threshold for the bit-flipping algorithm). To add noise, an initial number of neurons were picked uniformly at random and a random integer noise within the interval $[-z_{max}, z_{max}]$ (excluding 0) was added to each node. In Fig. 2, $z_{max} = 5$. For the simulated setting with $S = 15$, evaluating the lower bound in the proof of Theorem 2 yields that one is able to store $C_{max} = 1.5^{300} \approx 6.72 \times 10^{52}$ patterns of length $n = 600$!

Note that in order to do simulations, we simply generated a number of $m \times n$ random binary graphs H and did not check the expansion property. Hence, the performance illustrated in Fig. 2 is an upper bound on the actual probability of error. One might obtain better results by generating expander matrices or optimizing row and column degrees (d_c and d_p). Furthermore, increasing m and n will lead to better results even in the random case.

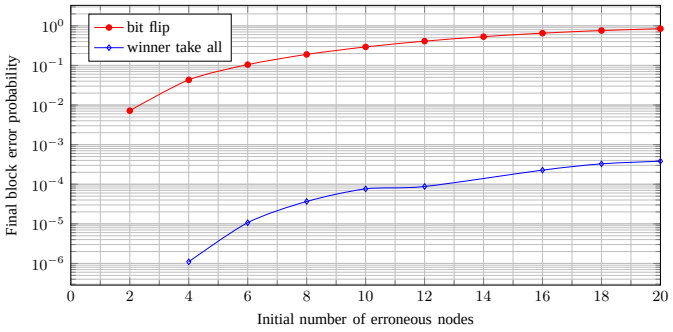

Fig. 2. Pattern retrieval error probability against the initial number of erroneous nodes for $n = 600$, $m = 300$, $d_c = 10$, $d_p = 5$ and a maximum noise magnitude of 5. Since bit-flip and winner-take-all algorithms are guaranteed to correct 1 and 2 initial errors, respectively, the corresponding block error probabilities are not illustrated in the figure due to the logarithmic scales.

Acknowledgment

The authors would like to thank Prof. Wulfram Gerstner for helpful comments and discussions. This work was supported by Grant 228021-ECCSciEng of the European Research Council.

References

[1] C. E. Shannon, *A mathematical theory of communication*, Bell Syst. tech. Journal, vol. 27, no. 379, pp. 623, 1948.

[2] W. A. McCulloch and W. Pits, *A logical calculus for ideas immanent in nervous activity*, Bulletin of Math. Biophys. 5, pp. 115 - 133, 1943.

[3] J. J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proc. Natl. Acad. Sci., Vol. 79, 1982, pp. 2554-2558.

[4] J. J. Hopfield, *Neurons with graded response have collective computational properties like those of two-state neurons*, Proc. Natl. Acad. Sci., Vol. 81, No. 10, 1984, pp. 3088 - 3092.

[5] 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.

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

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

[8] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer, 2000.

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

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

[11] W. Xu, B. Hassibi, *Efficient compressive sensing with deterministic guarantees using expander graphs*, Proc. Inf. Theo. Workshop (ITW), 2007, pp. 414-419.

[12] M. Sipser and D. Spielman, *Expander codes*, IEEE Trans. Inform. Theory, Vol. 42, No. 6, pp. 1710-1722, 1996.

[13] J. Komlos, R. Paturi, *Effect of connectivity in an associative memory model*, J. Computer and System Sciences, 1993, pp. 350-373.

[14] M. K. Muezzinoglu, C. Guzelis, J. M. Zurada, *A new design method for the complex-valued multistate Hopfield associative memory*, IEEE Trans. Neur. Net., Vol. 14, No. 4, 2003, pp. 891-899.

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

[16] A. H. Salavati, K. R. Kumar, W. Gerstner, A. Shokrollahi, *Neural Pre-coding Increases the Pattern Retrieval Capacity of Hopfield and Bidirectional Associative Memories*, To be presented at the IEEE Intl. Symp. Inform. Theory (ISIT - 11), St. Petersburg, Aug 2011.

[17] F.T. Sommer, G. Palm, *Improved bidirectional retrieval of sparse patterns stored by Hebbian learning*, Neural Networks, 1999, pp. 281-297.

[18] 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.

[19] D. Burshtein and G. Miller, *Expander graph arguments for message passing algorithms*, IEEE Trans. Inform. Theory, pp. 782-790, Feb. 2001.

[20] S. Jankowski, A. Lozowski, J.M., Zurada, *Complex-valued multistate neural associative memory*, IEEE Tran. Neur. Net., Vol. 1 , No. 6, 1996, pp. 1491-1496.

[21] D. L. Lee, *Improvements of complex-valued Hopfield associative memory by using generalized projection rules*, IEEE Tran. Neur. Net.,Vol. 12, No. 2, 2001, pp. 439-443.