Multi-Level Error-Resilient Neural Networks

Amir Hesam Salavati and Amin Karbasi

School of Computer and Communication Sciences, Ecole Polytechnique Federale de Lausanne (EPFL)

Email: hesam.salavati@epfl.ch, amin.karbasi@epfl.ch,

Abstract—The problem of neural network association is to retrieve a previously memorized pattern from its noisy version using a network of neurons. An ideal neural network should include three components simultaneously: a learning algorithm, a large pattern retrieval capacity and resilience against noise. Prior works in this area usually improve one or two aspects at the cost of the third.

Our work takes a step forward in closing this gap. More specifically, we show that by forcing natural constraints on the set of learning patterns, we can drastically improve the retrieval capacity of our neural network. Moreover, we devise a learning algorithm whose role is to learn those patterns satisfying the above mentioned constraints. Finally we show that our neural network can cope with a fair amount of noise.

I. INTRODUCTION

Neural networks are famous for their ability to *learn* and *reliably* perform a required task. An important example is the case of (associative) memory where we are asked to memorize (learn) a set of given patterns. Later, corrupted versions of the memorized patterns will be shown to us and we have to return the correct memorized patterns. In essence, this problem is very similar to the one faced in communication systems where the goal is to reliably transmit and efficiently decode a set of patterns (so called codewords) over a noisy channel.

As one would naturally expect, reliability is certainly a very important issue both in the neural associative memories and in communication systems. Indeed, the last three decades witnessed many reliable artificial associative neural networks. See for instance [5], [14], [6], [9], [13], [4].

However, despite common techniques and methods deployed in both fields (e.g., graphical models, iterative algorithms, etc), there has been a quantitative difference in terms of another important criterion: the efficiency. In communication systems, by using modern coding techniques it has become clear that the number of reliably transmitted codewords over a noisy channel can be made exponential in n, the length of the codewords. However, using current neural networks of size n to memorize a set of *randomly* chosen patterns, the maximum number of patterns that can be reliably memorized scales linearly in n [8], [14].

In search for the reasons beyond the inefficiency of the storage capacity of neural networks, we found out that a large body of past work (e.g., [5], [14], [6], [9]) followed a common assumption that a neural network should be able to memorize *any* subset of patterns drawn randomly from the set of all possible vectors of length n. Although this assumption gives the network a sense of generality, it reduces its storage capacity to a great extent.

An interesting question which arises in this context is whether one can increase the storage capacity of neural networks beyond the current linear scaling and achieve results similar to coding theory. To this end, Kumar et al. [7] suggested a new formulation of the problem where only a suitable set of patterns was considered for storing. This way they could show that the performance of neural networks in terms of storage capacity increases significantly. Following the same philosophy, we will focus on memorizing a random subset of patterns of length n such that the dimension of the training set is $k < n$. In other words, we are interested in memorizing a set of patterns that have a certain degree of *structure* and *redundancy*. We exploit this structure both to increase the number of patterns that can be memorized (from linear to exponential) and to increase the number of errors that can be corrected when the network is faced with corrupted inputs.

The success of [7] is mainly due to forming a bipartite network/graph (as opposed to a complete graph) whose role is to enforce the suitable constraints on the patterns, very similar to the role played by Tanner graphs in coding. More specifically, one layer is used to feed the patterns to the network (so called variable nodes in coding) and the other takes into account the inherent structure of the input patterns (so called check nodes in coding). A natural way to enforce structures on inputs is to assume that the connectivity matrix of the bipartite graph is orthogonal to all of the input patterns. However, the authors in [7] heavily rely on the fact that the bipartite graph is fully known and given, and satisfies some sparsity and expansion properties. The expansion assumption is made to ensure that the resulting set of patterns are resilient against fair amount of noise. Unfortunately, no algorithm for finding such a bipartite graph was proposed.

Our main contribution in this paper is to relax the above assumptions while achieving better error correction performance. More specifically, we first propose an iterative algorithm that can find a sparse bipartite graph that satisfies the desired set of constraints. We also provide an upper bound on the block error rate of the method that deploys this learning strategy. We then proceed to devise a multi-layer network whose performance in terms of error tolerance improves significantly upon [7] and no longer needs to be an expander.

II. PROBLEM FORMULATION

In contrast to the mainstream work in neural associative memories, we focus on non-binary neurons, i.e., neurons that can assume a finite set of integer values $\mathcal{S} = \{0, 1, \ldots, S-1\}$

for their states (where $S > 2$). A natural way to interpret the multi-level states is to think of the short-term (normalized) firing rate of a neuron as its output. Neurons can only perform simple operations. In particular, we restrict the operations at each neuron to a *linear summation* over the inputs, and a possibly *non-linear thresholding* operation. In particular, a neuron x updates its state based on the states of its neighbors $\{s_i\}_{i=1}^{n}$ as follows:

1) It 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.
2) It 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 $\mathcal{S}$.

Neural associative memory aims to memorize C patterns of length n by determining the weighted connectivity matrix of the neural network (*learning phase*) such that the given patterns are stable states of the network. Furthermore, the network should be able to tolerate a fair amount of noise so that it can return the correct memorized pattern in response to a corrupted query (*recall phase*). Among the networks with these two abilities, the one with largest C is the most desirable.

We first focus on learning the connectivity matrix of a neural graph which memorizes a set of patterns having some inherent redundancy. More specifically, we assume to have C vectors of length n with non-negative integer entries, where these patterns form a subspace of dimension $k < n$. We would like to memorize these patterns by finding a set of non-zero vectors $w_1, \ldots, w_m \in \mathbb{R}^n$ that are orthogonal to the set of given patterns. Furthermore, we are interested in rather sparse vectors. Putting the training patterns in a matrix $\mathcal{X}_{C \times n}$ and focusing on one such vector w, we can formulate the problem as:

$$\min \|\mathcal{X} \cdot w\|_2 \tag{1a}$$

subject to

$$\|w\|_0 \leq q \quad \text{and} \quad \|w\|_2^2 \geq \epsilon \tag{1b}$$

where $q \in \mathbb{N}$ determines the degree of sparsity and $\epsilon \in \mathbb{R}^+$ prevents the all-zero solution. A solution to the above problem yields a sparse bipartite graph which corresponds to the basis vectors of the null space specified by the patterns in the training set. It can therefore be described by Figure 1 with a connectivity matrix $W \in \mathbb{R}^{m \times n}$ such that $\mathcal{X}W^T = 0$.

In the recall phase, the neural network is fed with noisy inputs. A possibly noisy version of an input pattern is initialized as the states of the pattern neurons $x_1, x_2, \ldots, x_n$. Here, we assume that the noise is integer valued and additive (modulu S). In formula, we have $y = W(x^\mu + z) = Wz$ where z is the noise added to pattern x^μ. Hence, we are looking for an algorithm that can use this information to provably eliminate the effect of noise and return the correct pattern.

Remark 1. *A solution in the learning/recall phase is acceptable only if it can be found by simple operations at neurons.*

Before presenting our solution, we briefly overview the relation between the previous works and the one presented in this paper.
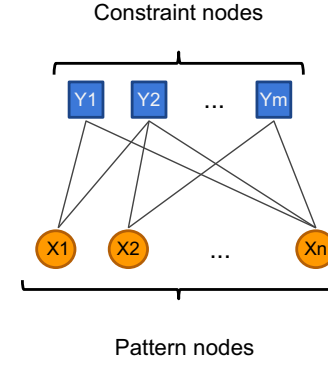

Fig. 1. A bipartite graph that represents the constraints on the training set.

A. Related Works

Designing artificial associative memories has been an active topic of research for the past three decades. Inspired by the Hebbian learning rule, Hopfield in his seminal work [5] introduced the Hopfield network: an auto-associative neural mechanism of size n with binary state neurons in which patterns are assumed to be binary vectors of length n. The capacity of a Hopfield network under vanishing block error probability was later shown to be $O(n/\log(n))$ in [8].

Due to the low capacity of Hopfield networks, extension of associative memories to non-binary neural models has also been explored in the past. For instance, in [6] the authors investigated a multi-state complex-valued neural associative memories for which the estimated capacity is $C < 0.15n$. Under the same model but using a different learning method, Muezzinoglu et al. [9] showed that the capacity can be increased to $C = n$. However the complexity of the weight computation mechanism is prohibitive.

A line of recent work has made considerable efforts to exploit the inherent structure of the patterns in order to increase both capacity and error correction capabilities. Such methods either make use of higher order correlations of patterns or focus merely on those patterns that have some sort of redundancy. As a result, they differ from previous methods for which every possible random set of patterns was considered. Pioneering this prospect, Berrou and Gripon [4] achieved considerable improvements in the pattern retrieval capacity of Hopfield networks, by utilizing Walsh-Hadamard sequences. Using low correlation sequences has also been considered in [13], which results in increasing the storage capacity of Hopfield networks to n without requiring any separate decoding stage.

In contrast to the pairwise correlation of the Hopfield model [5], Peretto et al. [11] deployed *higher order* neural models: the state of the neurons not only depends on the state of their neighbors, but also on the correlation among them. Under this model, they showed that the storage capacity of a higher-order Hopfield network can be improved to $C = O(n^{p-2})$, where p is the degree of correlation considered. The main drawback of this model was again the huge computational complexity required in the learning phase. To address this difficulty while being able to capture higher-order correlations, a bipartite

graph inspired from iterative coding theory was introduced in [7]. Under the assumptions that the bipartite graph is known, sparse, and expander, the proposed algorithm increased the pattern retrieval capacity to $C = O(a^n)$, for some $a > 1$. The main drawbacks in the proposed approach is the lack of a learning algorithm as well as the assumption that the weight matrix should be an expander. The sparsity criterion on the other hand, as it was noted by the authors, is necessary in the recall phase and biologically more meaningful.

In this paper, we focus on solving the above two problems in [7]. We start by proposing an iterative learning algorithm that identifies a *sparse* weight matrix W. The weight matrix W should satisfy a set of linear constraints $\mathcal{X}W^T = 0$. We then propose a novel network architecture which eliminates the need for the expansion criteria while achieving better performance than the error correction algorithm proposed in [7].

To learn a sparse neural graph, we follow ideas borrowed from iterative neural learning [15], [10] and compressive sensing [2], [3]. Constructing a factor-graph model for neural associative memory has been also addressed in [1] where a message-passing algorithm is proposed to memorize any set of random patterns. However, in this paper we focus on memorizing patterns belonging to subspaces with sparsity in mind as well. The difference would again be apparent in the pattern retrieval capacity (linear vs. exponential in network size).

III. LEARNING ALGORITHM

We are interested in an iterative algorithm that is simple enough to be implemented by a network of neurons. Therefore, we first relax (1) as follows:

$$\min \|\mathcal{X} \cdot w\|_2 - \lambda(\|w\|_2^2 - \epsilon) + \gamma(g(w) - q'). \quad (2)$$

In the above problem, we have approximated the constraint $\|w\|_0 \leq q$ with $g(w) \leq q'$ since $\|.\|_0$ is not a well-behaved function. The function $g(w)$ is chosen such that it favors sparsity. For instance one can pick $g(w)$ to be $\|.\|_1$, which leads to ℓ_1-norm minimizations. In this paper, we consider the function

$$g(w) = \sum_{i=1}^{n} \tanh(\sigma w_i^2)$$

where σ is chosen appropriately. By calculating the derivative of the objective function and primal-dual optimization techniques we obtain the following iterative algorithm for (2) (the details are tedious and left to our technical report [12]):

$$y(t) = \frac{\mathcal{X} \cdot w(t)}{\|\mathcal{X}\|_2} \quad (3a)$$

$$w(t+1) = (1 + 2\lambda_t)w(t) - 2\alpha_t \frac{\mathcal{X}^\top y(t)}{\|\mathcal{X}\|_2} - \gamma_t \nabla g(w) \quad (3b)$$

$$\lambda_{t+1} = \left[\lambda_t + \delta(\epsilon - \|w\|_2^2)\right] \quad (3c)$$

$$\gamma_{t+1} = \left[\gamma_t + \delta(g(w) - q')\right] \quad (3d)$$

Algorithm 1 Iterative Learning
Input: pattern matrix $\mathcal{X}$, stopping point p.
Output: w
  while $\|y(t)\|_{\max} > p$ **do**
    Compute $y(t) = \frac{\mathcal{X} \cdot w(t)}{\|\mathcal{X}\|_2}$.
    Update $w(t+1) = \eta \left((1 + 2\lambda_t)w(t) - 2\alpha_t \frac{\mathcal{X}^\top y(t)}{\|\mathcal{X}\|_2}\right)_{\theta_t}$.
    Update $\lambda_{t+1} = \left[\lambda_t + \gamma(\epsilon - \|w\|_2^2)\right]$.
    $t \leftarrow t + 1$.
  end while

---

where t denotes the iteration number, $\mathcal{X}^\top$ is the transpose of matrix $\mathcal{X}$, δ and α_t are small step sizes and $[\cdot]_+$ denotes $\max(\cdot, 0)$.

For our choice of $g(w)$, the ith entry of the function $f(w) = \nabla g(w)$, denoted by $f_i(w)$ reduces to $2\sigma w_i(1 - \tanh(\sigma w_i^2)^2)$. For very small values of w_i, $f_i(w) \simeq w_i$ and for large values of w_i, $f_i(w) \simeq 0$. Therefore, by looking at (3b) we see that the last term is pushing small values in $w(t+1)$ towards zero while leaving the larger values intact. Therefore, we remove the last term completely and enforce small entries to zero in each update which in turn enforces sparsity. The final iterative learning procedure is shown in Algorithm 1.

Here, θ_t is a positive threshold at iteration t and $\eta(.)_{\theta_t}$ is the *point-wise* soft-thresholding function given below:

$$\eta(u)_\theta = \begin{cases} u & \text{if } |u| > \theta, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Remark 2. *the above choice of soft-theresholding function is very similar to the one selected by Donoho et al. in [3] in order to recover a sparse signal from a set of measurements. The authors prove that their choice of soft-threshold function results in* optimal *sparsity-undersampling trade-off.*

The next theorem derives the necessary conditions on α_t, λ_t and θ_t such that Algorithm 1 converges to a sparse solution.

Theorem 1. *If $\theta_t \to 0$ as $t \to \infty$ and if $\lambda_t < a_{\min}/(a_{\max} - a_{\min})$, then there is a proper choice of α_t in every iteration t that ensures constant decrease in the objective function $\|\mathcal{X}.w(t)\|_{\max}$. Here $a_{\min} = \min_\mu \|x^\mu\|^2/\|\mathcal{X}\|^2$ and $a_{\max} = \max_\mu \|x^\mu\|^2/\|\mathcal{X}\|^2$. For $\lambda_t = 0$, i.e. $\|w(t)\|_2 \geq \epsilon$, picking $0 < \alpha_t < 1$ ensures gradual convergence.*

    Sketch of the proof: Let $E(t) = \|y(t)\|_{\max}$. We would like to show that $E(t+1) < E(t)$ for all iterations t. We can write $w(t+1) = w'(t) - \chi(w'(t); \theta_t)$ where $w'(t) = (1 + 2\lambda_t)w(t) - 2\alpha_t \frac{\mathcal{X}^\top y}{\|\mathcal{X}\|_2}$. We can then show that $E(t+1) \leq \|\frac{\mathcal{X}.w'(t)}{\|\mathcal{X}\|_2}\|_{\max} + \theta_t$. By expanding $\frac{\mathcal{X}.w'(t)}{\|\mathcal{X}\|}$ we will obtain $E(t+1) \leq \|D_t\|_{\max}E(t) + \theta_t$ where $D(t) = (1 + 2\lambda_t)I_{C \times C} - 2\alpha_t \frac{\mathcal{X}\mathcal{X}^T}{\|\mathcal{X}\|_2^2}$. Hence we need $\theta_t \to 0$ as $t \to \infty$ and ensuring $\|D_t\|_{\max} < 1$. The latter condition is satisfied if $\lambda_t \leq a_{\min}/(a_{\max} - a_{\min})$. For the detailed proof, we refer the interested readers to our technical report [12]. ∎

It must be noted that the above algorithm gives one possible solution to the learning problem (2), as there are multiple local
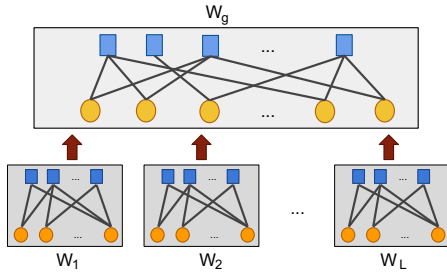
Fig. 2. A two-level error correcting neural network.

Algorithm 2 Error Correction

1: **for** $t = 1 \rightarrow t_{\max}$ **do**
2:    *Forward iteration:* Calculate the weighted input sum $h_i = \sum_{j=1}^{n} W_{ij} x_j$, for each neuron y_i and set:

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

3:    *Backward iteration:* Each neuron x_j computes

$$g_j = \frac{\sum_{i=1}^{m} W_{ij} y_i}{\sum_{i=1}^{m} |W_{ij}|}.$$

4:    Update the state of each pattern neuron j according to $x_j = x_j + \text{sgn}(g_j)$ only if $|g_j| > \varphi$.
5:    $t \leftarrow t + 1$
6: **end for**

---

minimums for this problem corresponding to different null-bases of the subspace defined by the patterns in the training set. Any of these vectors are acceptable for the proposed algorithm.

IV. MULTI-LEVEL NETWORK ARCHITECTURE

In order to have error correction capabilities we propose a new network structure (see Figure 2). To make the description clear and simple we only concentrate on a two-level neural network. However, the generalization of this idea is trivial and left to the reader.

The idea behind this new architecture is that we divide the input pattern of size n into L sub-patterns of length n/L. Now we feed each sub-pattern to a neural network which enforces m constraints[1] on the sub-pattern in order to correct the input errors. Such model might be specially useful in cases where the input is *modular*, similar to the case of memorizing different words of a sentence and enforcing global grammatical constraints on the sentence as a whole, or the case where local sub-patterns have few dominant principle components.

The local networks in the first level and the global network in the second level use Algorithm 2, which is a variant of the "bit-flipping" method proposed in [7], to correct the errors. Note 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 $y_i = 0$. The quantity g_j can be interpreted

---

[1]The number of constraints for different networks can vary. For simplicity of notifications we assume equal sizes.

as feedback to pattern neuron x_j from the constraint neurons. Hence, 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.

Theorem 2. *Let $\bar{d}$ and d_{min} be the average and minimum degree of pattern neurons, respectively. Then, Algorithm 2 can correct a single error in the input pattern with probability at least $1 - \left(\frac{\bar{d}}{m}\right)^{d_{min}}$ if $\varphi \rightarrow 1$.*

For the proof, we refer the interested readers to our technical report [12].

Given that each local network is able to correct one pattern with high probability, L such networks can correct L input errors *if* they are separated such that only one error appears in the input of each local network. Otherwise, there would be a probability that the network could not handle the errors. In that case, we feed the overall pattern of length n to the second layer with the connectivity matrix W_g, which enforces m_g *global* constraints. And since the probability of correcting two erroneous nodes increases with the input size, we expect to have a better error correction probability in the second layer. Therefore, using this simple scheme we expect to gain a lot in correcting errors in the patterns. In the next section, we provide simulation results which confirm our expectations and show that the block error rate can be improved by a factor of 100 in some cases.

Remark 3. *The number of constraints for the second layer affects the gain one obtains in the error performance. Intuitively, if the number of global constraints is large, we are enforcing more constraints so we expect a better error performance. We can think of determining the number even adaptively, i.e. if the error performance that we are getting is unacceptable, we can look deeper in patterns to identify their internal structure by searching for more constraints. This would be a subject of our future research.*

V. PATTERN RETRIEVAL CAPACITY

The following theorem shows that the proposed neural architecture is capable of memorizing an exponential number of patterns in n.

Theorem 3. *Let $\mathcal{X}$ be the $C \times n$ matrix, formed by C vectors of length n with non-negative integer entries between 0 and $S - 1$. Furthermore, let $k_g = rn$ for some $0 < r < 1$. Then, there exists a data set $\mathcal{X}$ with $C = a^{rn}$, $a > 1$, and $\text{rank}(\mathcal{X}) = k_g < n$, such that they can be memorized by the proposed multi-level neural network shown in figure 2.*

The proof of the theorem can be found in the technical report [12].

VI. SIMULATION RESULTS

We have simulated the proposed learning algorithm in the multi-level architecture to investigate the block error rate of the suggested approach and the gain we obtain in error rates by adding a second level. We constructed 4 local networks, each with $n/4$ pattern and m constraint nodes.

A. Learning Phase

We generated a sample data set of $C = 10000$ patterns of length n where each block of $n/4$ belonged to a subspace of dimension $k < n/4$. Note that C can be an exponential number in n. However, we selected $C = 10000$ as an example to show the performance of the algorithm because even for small values of k, and exponential number in k will become too large to handle numerically. The result of the learning algorithm is four different local connectivity matrices $W_1, \ldots, W_4$ as well as a global weight matrix W_g. The number of local constraints was $m = n/4 - k$ and the number of global constraints was $m_g = n - k_g$, where k_g is dimension of the subspace for overall pattern.

Table VI-A shows the average number of iterations executed before convergence is reached for different local and global constraints. It also gives the average sparsity ratio for the columns of weight matrices. The sparsity ratio is defined as $\rho = \kappa/n$, where κ is the number of non-zero elements.

TABLE I
AVERAGE NUMBER OF CONVERGENCE ITERATIONS AND SPARSITY IN THE LOCAL AND GLOBAL NETWORKS FOR $n = 400$

	Sparsity Ratio		Convergence Steps	
	$k_g = 100$	$k_g = 200$	$k_g = 100$	$k_g = 200$
Local	0.28	0.32	4808	5064
Global	0.22	0.26	14426	33206

B. Recall Phase

For the recall phase, in each trial we pick a pattern randomly from the training set, corrupt a given number of its symbols with ± 1 noise and use the suggested algorithm to correct the errors. As mentioned earlier, the errors are corrected first at the local and the at the global level. When finished, we compare the output of the first and the second level with the original (uncorrupted) pattern x. An error is declared if the output does not match at each stage.

Figure 3 illustrates the pattern error rates $n = 400$ with two different values of $k_g = 100$ and $k_g = 200$. The results are also compared to that of the bit-flipping algorithm in [7] to show the improved performance of the proposed algorithm. As one can see, having a larger number of constraints at the global level, i.e. having a smaller k_g, will result in better pattern error rates at the end of the second stage.

Table VI-B shows the gain we obtain by adding an additional second level to the network architecture. The gain is calculated as the ratio between the pattern error rate at the output of the first and the second level.

TABLE II
GAIN IN PATTERN ERROR RATE (PER) FOR DIFFERENT VALUES OF $n = 400$ AND INITIAL NUMBER OF ERRORS

Number of initial errors	Gain for $k_g = 100$	Gain for $k_g = 200$
2	10.2	2.79
3	6.22	2.17
4	4.58	1.88
5	3.55	1.68
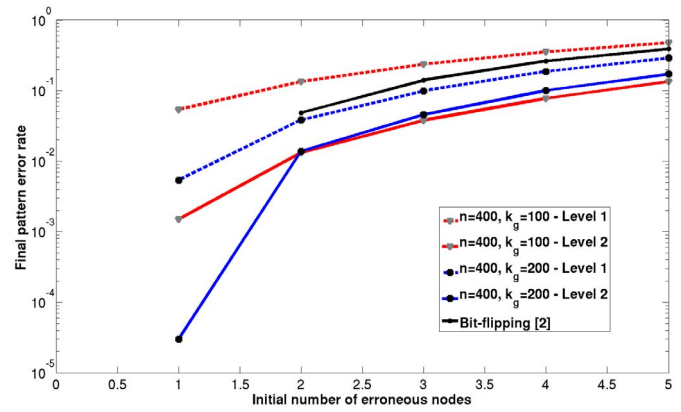
Fig. 3. Pattern error rate against the initial number of erroneous nodes

REFERENCES

[1] A. Braunstein, R. Zecchina, *Learning by message-passing in networks of discrete synapses*, Phys. Rev. Lett., Vol. 96, No. 3, 2006, pp. 030201-1-030201-4

[2] E. Cands, T. Tao, *Near optimal signal recovery from random projections: Universal encoding strategies?*, IEEE Trans. on Information Theory, Vol. 52, No. 12, 2006, pp. 5406-5425.

[3] D. L. Donoho, A. Maleki, A. Montanari, *Message passing algorithms for compressed sensing*, Proc. Nat. Acad. Sci., Vol. 106, 2009, pp. 1891418919.

[4] V. Gripon, C. Berrou, *Sparse neural networks with large learning diversity*, IEEE Trans. on Neural Networks, Vol. 22, No. 7, 2011, pp. 10871096.

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

[6] 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.

[7] K.R. Kumar, A.H. Salavati and A. Shokrollahi, *Exponential pattern retrieval capacity with non-binary associative memory*, Proc. IEEE Int. Theory Work., 2011, pp. 80-84.

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

[9] 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.

[10] E. Oja, T. Kohonen, *The subspace learning algorithm as a formalism for pattern recognition and neural networks*, Neural Networks, Vol. 1, 1988, pp. 277-284.

[11] P. Peretto, J. J. Niez, *Long term memory storage capacity of multiconnected neural networks*, Biological Cybernetics, Vol. 54, No. 1, 1986, pp. 53-63.

[12] A. H. Salavati, A. Karbasi, Multi-Level Error-Resilient Neural Networks with Learning, arXiv:1202.2770v1 [cs.NE]

[13] A. H. Salavati, K. R. Kumar, W. Gerstner, A. Shokrollahi, *Neural Pre-coding Increases the Pattern Retrieval Capacity of Hopfield and Bidirectional Associative Memories*, IEEE Intl. Symp. Inform. Theory (ISIT-11), 2011, pp. 850-854.

[14] 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.

[15] L. Xu, A. Krzyzak, E. Oja, Neural nets for dual subspace pattern recognition method, Int. J. Neur. Syst., Vol. 2, No. 3, 1991, pp. 169-184.