Iterative Learning and Denoising in Convolutional Neural Associative Memories

Amin Karbasi AMIN.KARBASI@EPFL.CH
Information Processing Group, Ecole Polytechnique Federale de Lausanne (EPFL), 1015 Lausanne, Switzerland

Amir Hesam Salavati HESAM.SALAVATI@EPFL.CH
Algorithms Laboratory, Ecole Polytechnique Federale de Lausanne (EPFL), 1015 Lausanne, Switzerland

Amin Shokrollahi AMIN.SHOKROLLAHI@EPFL.CH
Algorithms Laboratory, Ecole Polytechnique Federale de Lausanne (EPFL), 1015 Lausanne, Switzerland

Abstract

The task of a neural associative memory is to retrieve a set of previously memorized patterns from their noisy versions by using a network of neurons. Hence, an ideal network should be able to 1) gradually learn a set of patterns, 2) retrieve the correct pattern from noisy queries and 3) maximize the number of memorized patterns while maintaining the reliability in responding to queries. We show that by considering the inherent redundancy in the memorized patterns, one can obtain all the mentioned properties at once. This is in sharp contrast with previous work that could only improve one or two aspects at the expense of the others. More specifically, we devise an iterative algorithm that learns the redundancy among the patterns. The resulting network has a retrieval capacity that is exponential in the size of the network. Lastly, by considering the local structures of the network, the asymptotic error correction performance can be made linear in the size of the network.

1. Introduction

The ability to memorize a large set of patterns and reliably retrieve them in the presence of noise, are among the main reasons that attracted a large body of research on neural networks for the past three decades. Ideally, a perfect neural associative memory should be able to *learn* patterns, have a large pattern retrieval *capacity* and be *noise-tolerant*. This problem, called "associative memory", is in spirit very similar to reliable information transmission faced in communication systems where the goal is to efficiently decode a set of transmitted patterns over a noisy channel.

Despite this similarity and common methods deployed in both fields (e.g., graphical models, iterative algorithms, to name a few), there is a huge gap between the efficiency achieved by them. More specifically, by deploying modern coding techniques, it was shown that the number of reliably transmitted patterns over a noisy channel can be made *exponential* in n, the length of the patterns. This was achieved by intelligently imposing redundancy among transmitted patterns. In contrast, the maximum number of patterns that can be reliably memorized by most current neural networks scales *linearly* in the size of the network. This is due to the 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 (see, for example, (Hopfield, 1982), (Venkatesh & Psaltis, 1989), (Jankowski et al., 1996), (Muezzinoglu et al., 2003)).

Recently, (Kumar et al., 2011) suggested a new formulation of the problem where only a suitable set of patterns was considered for storing. To enforce the set of constraints, they formed a bipartite graph (as opposed to a complete graph considered in the earlier work) where one layer feeds the patterns to the network and the other takes into account the inherent structure. The role of bipartite graph is indeed similar to the Tanner graphs used in modern coding techniques (Tanner). Using this model, (Kumar et al., 2011) provided evidence that the resulting network can

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

potentially memorize a substantial number of patterns at the expense of correcting only a *single* error during the recall phase. Later, (Salavati & Karbasi, 2012) further improved this result by introducing a multi-layer structure that can correct a *constant* number of errors.

In this work, we introduce a novel neural architecture equipped with an online learning algorithm. This architecture can correct a *linear* fraction of errors while keeping the storage capacity *exponential* in the size of the network. This is achieved by memorizing a set of patterns with some degrees of redundancy, i.e., those with very weak minor components. In other words, we find dual vectors that are (almost) orthogonal to particular parts of input patterns. By making use of this inherent redundancy, we can increase the number of memorized patterns from linear to exponential. Our learning algorithm is an extension of the subspace learning method proposed by (Oja & Kohonen, 1988), with an additional property to make the learned vectors *sparse*. The sparsity property will become helpful during the noise-elimination phase.

We will provide theoretical analysis to support our claims and also assess the accuracy of our results through simulations. We evaluate the performance of our proposed algorithms over synthetic datasets, and compare them with prior art. We also investigate the effect of different hyper-parameters on the performance of the proposed method.

2. Related Work

Arguably, the Hopfield network is the first auto-associative neural mechanism capable of learning a set of patterns and recalling them later (Hopfield, 1982). By utilizing the Hebbian learning rule, Hopfield considered a neural network of size n with binary state neurons. It was shown by McEliece et al. (1987) that the capacity of a Hopfield network with n nodes is bounded by $\mathcal{C} = (n/2\log(n))$.

With the hope of increasing the capacity of the Hopfield network, an extension of associative memories to non-binary states has also been explored in the past. In particular, Jankowski et al. (1996) investigated a complex-valued neural associative memory where each neuron can be assigned to a multivalued state from the set of complex numbers. It was shown by Muezzinoglu et al. (2003) that the capacity of such networks can be increased to $\mathcal{C} = n$ at the cost of a prohibitively complex weight computation mechanism.

Recently, in order to increase the capacity and robustness, a line of work considered exploiting the inherent structure of the patterns. This is done by either making use of the correlations among the pattern or memorizing only those patterns that have some sort of redundancy. Note that this line of work differs from previous methods in one important aspect: not any possible set of patterns is considered for learning. By utilizing neural cliques, (Gripon & Berrou, 2011) were among the first who demonstrated that considerable improvements in the pattern retrieval capacity of Hopfield networks is possible, albeit still not passing the polynomial bound on the capacity, i.e. $\mathcal{C} = O(n^2)$.

By deploying higher order neural models, rather than pairwise correlation considered in Hopfield networks, (Peretto & Niez, 1986) showed that the storage capacity can be improved to $\mathcal{C} = O(n^{p-2})$, where p is the degree of correlation. In such models, the state of the neurons not only depends on the state of their neighbors, but also on the correlations among them. However, the main drawback of this work lies in the prohibitive computational complexity of the learning phase. To address this difficulty, while being able to capture higher-order correlations, a new model based on bipartite graphs was introduced by (Kumar et al., 2011), and was further explored by (Salavati & Karbasi, 2012). Under the restrictive assumptions that the bipartite graph is fully known, sparse, and expander, the proposed algorithm by (Kumar et al., 2011) increased the pattern retrieval capacity to $\mathcal{C} = O(a^n)$, for some $a > 1$. The lack of a learning algorithm that satisfies all the mentioned requirements are among the reasons that limits the capabilities of this model.

In this paper, we propose a novel architecture to capture the internal redundancy by dividing the input patterns into *overlapping* clusters/patches. We first devise an online learning algorithm that effectively learns the structure of the bipartite graph and show that the capacity of the resulting network is exponential in its size. We then propose a novel error correction algorithm that corrects a linear fraction of errors in the recall phase. Our analytical results, supported by simulations, demonstrate that introducing overlaps among clusters improves the error correction performance significantly.

It is worth mentioning that learning a set of input patterns with robustness against noise is not just the focus of neural associative memory. For instance, (Vincent et al., 2008) proposed an interesting approach to extract robust features in autoencoders. Their approach is based on artificially introducing noise during the learning phase and let the network learn the mapping between the corrupted input and the correct version. This way, they shift the burden from the recall phase to the learning phase. We, on the other hand, consider a

more particular form of redundancy and enforce a suitable structure which helps us design algorithms that are faster and *guaranteed* to correct a linear fraction of noise without previously being exposed to.

We should also mention that learning linear constraints by a neural network is hardly a new topic. It is shown by (Xu et al., 1991) and (Oja & Kohonen, 1988) that one can learn a matrix orthogonal to a set of patterns in the training set by using simple neural learning rules. However, to our knowledge, finding such a matrix subject to sparsity has not been addressed before.

Deep Belief Networks: Our neural architecture is in some aspects similar to those of Convolutional (Deep) Belief Networks. Deep Belief Networks (DBNs) are typically used to extract/classify features by means of several consecutive stages (e.g., pooling, rectification, etc). Having multiple stages help the network to learn more interesting and complex features. An important class of DBNs are convolutional DBNs where the input layer (or the receptive field) is divided into multiple possibly overlapping patches, and the network extract features from each patch (Jarrett et al., 2009).

Since we divide the input patterns into a few overlapping smaller clusters, our model is similar to that of convolutional DBNs. Furthermore, we also learn multiple *features* (i.e., dual vectors) from each patch where the feature extractions differ over different patches. This is indeed very similar to (Le et al., 2010).

In contrast to convolutional DBNs, the focus of this work is not classification but rather recognition of the *exact* patterns from their noisy versions. Moreover, in most DBNs, we not only have to find the proper dictionary for classification, but we also need to calculate the features for each input pattern. This alone increases the complexity of the whole system, especially if denoising is part of the objective. In our model, however, the dictionary is defined in terms of dual vectors. Consequently, previously memorized patterns are computationally easy to recognize as they yield the all-zero vector in the output of the feature extraction stage. In other words, a non-zero output can only happen if the input pattern is noisy. Another advantage of our model over DBNs is a much faster learning phase. More precisely, by using a single overlapping layer in our model the information diffuses gradually in the network. The same criterion is achieved in DBNs (Socher et al., 2011) by constructing several stages.

3. Notation and Definitions

In neural associative memories - the subject of this work - the goal is to design a neural network capable of memorizing a large set of patterns from a data set $\mathcal{X}$ (learning phase), and recalling them later in presence of noise (recall phase).

Learning phase: Each pattern $x = (x_1, x_2, \dots, x_n)$ is a vector of length n, where $x_i < \mathcal{S} = \{0, \dots, S-1\}$ for $i \in [n]$ and some integer S. In this work, our focus is on memorizing patterns with strong *local correlation* among the entries. More specifically, we divide the entries of each pattern x into L *overlapping* sub-patterns of lengths $n_1, \dots, n_L$, so that $\sum n_i \geq n$. Note that due to overlaps, a pattern node can be a member of multiple sub-patterns, as shown in Figure 1. We denote the i-th sub-pattern by $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_{n_i}^{(i)})$. To enforce local correlations, we assume that the sub-patterns $x^{(i)}$ form a subspace of dimension $k_i < n_i$. This is done by imposing linear constraints on each cluster. These linear constraints are captured in the form of dual vectors as follows. In the learning phase, we would like to memorize these patterns by finding a set of non-zero vectors $w_1^{(i)}, w_2^{(i)}, \dots, w_{m_i}^{(i)}$ that are orthogonal to the set of sub-patterns $x^{(i)}$, i.e.,

$$y_j^{(i)} = (w_j^{(i)})^\top \cdot x^{(i)} = 0, \quad \forall j \in [m_i] \, \forall i \in [L], \qquad (1)$$

where $[q]$ denotes the set $\{1, 2, \cdots, q\}$. The weight matrix $W^{(i)} = [w_1^{(i)} | w_2^{(i)} | \dots | w_{m_i}^{(i)}]^\top$ of cluster i is created by putting all the dual vectors next to each other. Equation (1) can be written equivalently as $W^{(i)} \cdot x^{(i)} = 0$. Therefore, the goal is to devise a low complexity algorithm that updates the weights of $W^{(i)}$'s (for $i \in [L]$) once it encounters a new pattern $x \in \mathcal{X}$. We should stress here that contrary to our iterative algorithm, many proposed learning algorithms in the literature can learn a set of patterns only after the whole set is presented to the neural network. Consequently, they are usually unable to learn a new set of patterns. Removing this restriction is one of the major contributions of our work.

One can easily map the local constraints imposed by the $W^{(i)}$'s into a global constraint by introducing a global weight matrix W of size $m \times n$. The first m_1 rows of the matrix W correspond to the constraints in the first cluster, rows $m_1 + 1$ to $m_1 + m_2$ correspond to the constraints in the second cluster, and so forth. Hence, by inserting zero entries at proper positions, we can construct the global constraint matrix W. We will use both the local and global connectivity matrices to eliminate noise in the recall phase.

Recall phase: Once the set of patterns $\mathcal{X}$ has been

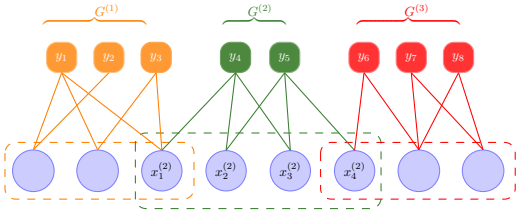

Figure 1: Bipartite graph G.

memorized, a desirable neural network should be able to retrieve an already memorized pattern from its corrupted version (of course, if the intensity of noise is not moderate, there is no hope for recovery). Needless to say, the recall process should be implementable by simple operations mentioned earlier.

In the recall phase a noisy version, say y, of an already learned pattern $x \in \mathcal{X}$ is given. Here, we assume that the noise is an additive vector of size n, denoted by e, whose entries assume values independently from $\{-1, 0, +1\}$ with corresponding probabilities $p_{-1} = p_{+1} = p_e/2$ and $p_0 = 1 - p_e$.[1] To ensure that the amount of noise is moderate, we also need to assume that the error probability, p_e, is less than p_0. We denote by $e^{(i)}$, the realization of noise on the subpattern $x^{(i)}$. In formula, $y = x + e$ (modulo S). Note that $W \cdot y = W \cdot e$ and $W^{(i)} \cdot y^{(i)} = W^{(i)} \cdot e^{(i)}$. Therefore, the goal will be to remove the noise e and obtain the desired pattern x as the true states of the pattern neurons. This task will be accomplished by exploiting the fact that we have chosen the set of patterns $\mathcal{X}$ to satisfy the set of constraints $W^{(i)} \cdot x^{(i)} = 0$.

Capacity: The last issue we look at in this work is the retrieval capacity $\mathcal{C}$ of our proposed method. Formally, the retrieval capacity is defined in terms of the maximum number of patterns that a neural network can learn and distinguish later. By construction, we show that the retrieval capacity of our network is exponential in the size of the network.

4. The Learning Algorithm

In this section, we discuss the learning algorithm for a given cluster ℓ. The case of other clusters will be a straightforward extension of the suggested algorithm. Since the patterns lie in a subspace of dimension $k_\ell \leq n_\ell$, we adapt the algorithm proposed in (Oja & Karhunen, 1985) and (Xu et al., 1991) to learn the dual space of the subspace defined by the patterns. Due to requirements of the denoising algorithm used in the recall phase, we also need the dual vectors to be sparse. To this end, we add an additional term to penalize non-sparse solutions during the learning phase. Furthermore, we are not looking for an orthogonal basis as in (Xu et al., 1991). Instead, we would like to find m_ℓ vectors that are orthogonal to the patterns. The problem is to find a constraint *vector* $w^{(\ell)}$ is given by

$$\min_{w^{(\ell)}} \sum_{x^{(\ell)} \subset x \in \mathcal{X}} |(x^{(\ell)})^\top \cdot w^{(\ell)}|^2 + \beta g(w^{(\ell)}), \quad \text{s.t. } \|w^{(\ell)}\|_2 = 1. \tag{2}$$

In the above problem, $x \in \mathcal{X}$ is a pattern drawn from the training set, β is a positive constant and $g(.)$ is the penalty term to favor sparse results. For instance one can pick $g(w^{(\ell)}) = \|w^{(\ell)}\|_1$, which leads to the ℓ_1-norm penalty, widely used in compressed sensing (Donoho, 2006), (Candès & Tao, 2006). In this paper, however, we find it more appropriate to consider

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

Intuitively, for large σ, $\tanh(\sigma(w_i^{(\ell)})^2)$ approximates $|\text{sign}(w_i^{(\ell)})|$. Therefore, the larger σ gets, the closer $g(w^{(\ell)})$ will be to $\|\cdot\|_0$. By calculating the derivative of the objective function, and by considering the update required for each randomly picked pattern x, we will obtain the following iterative algorithm:

$$y^{(\ell)}(t) = x^{(\ell)}(t) \cdot w^{(\ell)}(t), \tag{3}$$

$$\tilde{w}^{(\ell)}(t+1) = w^{(\ell)}(t) - \alpha_t \Big(2y^{(\ell)}(t)x^{(\ell)}(t) + \beta\Gamma(w^{(\ell)}(t))\Big) \tag{4}$$

$$w^{(\ell)}(t+1) = \frac{\tilde{w}^{(\ell)}(t+1)}{\|\tilde{w}^{(\ell)}(t+1)\|_2}. \tag{5}$$

where t is the iteration number, $x^{(\ell)}(t)$ is the subpattern of a pattern $x(t)$ drawn at iteration t, α_t is a small positive constant and $\Gamma(w^{(\ell)}) = \nabla g(w^{(\ell)})$ is the gradient of the penalty term. This function has the interesting property that it suppresses very small values of $w_i^{(\ell)}(t)$. To see why, consider the i-th entry of $\Gamma(w^{(\ell)}(t)))$, namely,

$$\Gamma_i(w^{(\ell)}(t)) = \partial g(w^{(\ell)}(t))/\partial w_i^{(\ell)}(t) = 2\sigma_t w_i^{(\ell)}(t)(1 - \tanh^2(\sigma w_i^{(\ell)}(t)^2)).$$

It is easy to see that $\Gamma_i(w(t)) \simeq 2\sigma w_i(t)$ for relatively small values of $w_i(t)$, and $\Gamma_i(w^{(\ell)}(t)) \simeq 0$ for larger values of $w_i^{(\ell)}(t)$. Thus, for proper choices of β and σ, Eq (4) suppresses small entries of $w^{(\ell)}(t)$ towards zero and favors sparser results. To simplify the analysis, with some abuse of notation, we approximate the

[1] The rationale behind this noise model is to capture the event of neurons mis-fire a spike (+1 noise) or miss one (−1 noise).

Algorithm 1 Iterative Learning

Input: Dataset $\mathcal{X}$ with $|\mathcal{X}| = \mathcal{C}$, stopping point ε.
Output: $w^{(\ell)}$

```
while ∑_{x∈X} |(x^(ℓ))^⊤ · w^(ℓ)(t)|^2 > ε do
    Choose pattern x(t) uniformly at from X.
    Compute y^(ℓ)(t) = (x^(ℓ)(t))^⊤ · w(t).
    Update w^(ℓ)(t) according to Eq (8).
    t ← t + 1.
end while
```

function $\Gamma(w^{(\ell)}(t))$ with the following function:

$$\Gamma_i(w^{(\ell)}(t)) = \begin{cases} w_i^{(\ell)}(t) & \text{if } |w_i^{(\ell)}(t)| \leq \theta_t; \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Here θ_t is a small positive threshold.

Following the same approach as in (Oja & Karhunen, 1985) we assume α_t to be small enough such that Eq (5) can be expanded as powers of α_t. This leads to a simpler version of equations (3-5) as follows.

$$y^{(\ell)}(t) = x^{(\ell)}(t) \cdot w^{(\ell)}(t), \quad (7)$$

$$w^{(\ell)}(t+1) = w^{(\ell)}(t) - \alpha_t (y^{(\ell)}(t)(x^{(\ell)}(t) - \frac{y^{(\ell)}(t) w^{(\ell)}(t)}{\|w^{(\ell)}(t)\|_2^2}) + \beta \Gamma(w^{(\ell)}(t))). \quad (8)$$

In the above approximation, we also omitted the term $\alpha_t \beta \left(w^{(\ell)}(t) \cdot \Gamma(w^{(\ell)}(t))\right) w^{(\ell)}(t)$ since the contribution of $w^{(\ell)}(t) \cdot \Gamma(w^{(\ell)}(t))$ is negligible.

The resulting learning algorithm for one constraint node is shown in Algorithm 1. In words, $y^{(\ell)}(t)$ is the projection of $x^{(\ell)}(t)$ onto $w^{(\ell)}(t)$. If for a given data vector $x^{(\ell)}(t)$, the projection $y^{(\ell)}(t)$ is non-zero, then the weight vector will be updated in order to reduce this projection.

To prove the convergence of Algorithm 1, we benefit from the convergence of Stochastic Gradient Descent (SGD) algorithms (Bottou, 1998). Let $E(w^{(\ell)}) = \sum_{x \in \mathcal{X}} |x^{(\ell)} \cdot w^{(\ell)}|^2$ denote the cost function. Furthermore, let $A_x = x^{(\ell)}(x^{(\ell)})^\top$, and $A = \mathbb{E}\{A_x | x \in \mathcal{X}\}$ represent the correlation among patterns in the training set $\mathcal{X}$. Since patterns are uniformly sampled from $\mathcal{X}$, one can rewrite $E(w^{(\ell)}) = (w^{(\ell)})^\top A w^{(\ell)} / \mathcal{C}$.

Theorem 1 *Under the following regularity conditions*

- *[A1.]* $\|A\|_2 \leq \Upsilon < \infty$,
- *[A2.]* $\sup_{x \in \mathcal{X}} \|A_x\|_2 \leq \zeta < \infty$,
- *[A3.]* $\alpha_t \geq 0$, $\sum \alpha_t = \infty$,
- *[A4.]* $\sum \alpha_t^2 < \infty$,

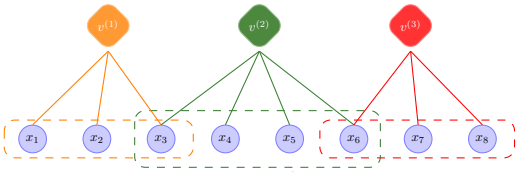

Figure 2: Contraction graph $\widetilde{G}$ corresponding to graph G in Figure 1.

- *[A5.] at each iteration* t, $2\alpha_t \beta < 1$,

Algorithm 1 converges to a local minimum $\hat{w}^{(\ell)}$ *for which* $\nabla E(\hat{w}^{(\ell)}) = 0$. *Furthermore, at this local minimum, the vector* $\hat{w}^{(\ell)}$ *is orthogonal to all the patterns of the training set, i.e.* $A\hat{w}^{(\ell)} = 0$.

The proof uses the results of (Bottou, 1998) to show the convergence to a local minimum proving that the weight vector at this local minimum is orthogonal to the patterns of the training set. Furthermore, condition A5 ensures that Algorithm 1 does not converge to the trivial solution $\hat{w}^{(\ell)} = 0$. The full proof could be found in the supplementary materials.

In order to find m_ℓ constraints required by the learning phase, we need to run Algorithm 1 several times. In practice, we can perform this process in parallel, to speed up the learning phase. It is also more meaningful from a biological point of view, as each constraint neuron can act independently from the others. Although running Algorithm 1 in parallel may result in redundant constraints, our experimental results show that by starting from different random initial points, the algorithm converges to linearly independent constraints.

5. Recall Phase

The recall phase of our proposed method consists of two parts, intra-module and inter-module. In the intra-module part, each cluster tries to remove noise from its sub-pattern. As we will show in Section 5.1, this is indeed possible if the sub-patterns experience a single error. In case of successful decoding, the pattern neurons keep their states and revert back to their original states otherwise. During the inter-module decoding, once the correction in cluster ℓ finishes, cluster $\ell + 1$ starts and this process continues in several rounds. Here, the overlapping structure of the clusters helps them correct a linear fraction of errors together.

5.1. Intra-Module Recall Algorithm

In the intra-module part, we exploit the fact that the connectivity matrix of the neural network in each clus-

ter is sparse and orthogonal to the memorized patterns. For a cluster ℓ, let $x^{(\ell)} + e^{(\ell)}$ be the given noisy input pattern. Recall that $W^{(\ell)}(x^{(\ell)}+e^{(\ell)}) = W^{(\ell)}e^{(\ell)}$.

Algorithm 2 performs a series of forward and backward iterations to remove $e^{(\ell)}$. At each iteration, the pattern neurons decide locally whether to update their current state or not: if the amount of feedback received by a pattern neuron exceeds a threshold, the neuron updates its state, and remains intact, otherwise.[2]

Theorem 2 *Algorithm 2 corrects at least a single error of cluster $G^{(\ell)}$ with probability at least $1 - \left(\frac{d_{avg}^{(\ell)}}{m}\right)^{d_{\min}^{(\ell)}}$ as $\varphi \to 1$, where $d_{avg}^{(\ell)}$ and $d_{\min}^{(\ell)}$ are the average and minimum pattern-node degrees, respectively.*

The proof can be found in the supplementary material. In short, we bound the probability of correcting a single error, P_{c_1}, in terms of the probability that two nodes share the same neighborhood in a cluster.

Theorem 2 implies that to have one error corrected with high probability, we have to make sure to have small average and large minimum degrees among the pattern nodes. To simplify the analysis of Section 5.2, we make a conservative assumption that a cluster is capable of correcting only a single error with overwhelming probability, and declares failure in case of multiple errors. In practice, as it is confirmed by our simulations, clusters are able to correct more than one error.

As stated earlier, the efficiency of Algorithm 2 relies on the assumption that the neural network is sparse. To gain some insight, consider an extreme case where the bipartite graph is complete. Then, a single error results in the violation of all constraint neurons in the forward iteration. Therefore, in the backward iteration, all the pattern neurons receive feedback from their neighbors. This makes it impossible to tell which pattern neuron is the noisy one. On the other hand, once the graph is sparse, a single error makes only a small set of constraint neurons unsatisfied. Consequently, in the backward iteration, only the pattern nodes that share the same neighborhood with the noisy one receive feedback. Note that in this case, the fraction of the received feedback will be much larger for the true noisy neuron. Therefore, by merely looking at the fraction of received feedback from the constraint neurons, one can identify the noisy pattern neuron.

[2]In order to maintain the current value of a neuron, we can add self-loops to pattern neurons in Figure 1 (the self-loops are not shown in the figure for the sake of clarity).

Algorithm 2 Intra-Module Error Correction

Input: Training set $\mathcal{X}$, threshold φ, iteration $t_{\max}$
Output: $x_1^{(\ell)}, x_2^{(\ell)}, \ldots, x_{n_\ell}^{(\ell)}$

1: **for** $t = 1 \to t_{\max}$ **do**
2: *Forward iteration:* Calculate the weighted input sum $h_i^{(\ell)} = \sum_{j=1}^{n_\ell} W_{ij}^{(\ell)} x_j^{(\ell)}$, for each neuron $y_i^{(\ell)}$ and set $y_i^{(\ell)} = \text{sign}(h_i^{(\ell)})$.
3: *Backward iteration:* Each neuron $x_j^{(\ell)}$ computes

$$g_j^{(\ell)} = \frac{\sum_{i=1}^{m_\ell} W_{ij}^{(\ell)} y_i^{(\ell)}}{\sum_{i=1}^{m_\ell} |W_{ij}^{(\ell)}|}.$$

4: Update the state of each pattern neuron j according to

$$x_j^{(\ell)} = x_j^{(\ell)} + \text{sign}(g_j^{(\ell)})$$

only if $|g_j^{(\ell)}| > \varphi$.
5: $t \leftarrow t + 1$
6: **end for**

5.2. Inter-Module Recall Algorithm

In the previous section, we showed that a single error inside a cluster can be corrected with high probability. In fact, a finer analysis reveals that there exists a threshold $T_e > 1$ such that any number of errors less than T_e can be corrected. Since clusters have overlaps with one another, removing an error from a cluster, can potentially help the others.

Our proposed algorithm is based on a famous error correction decoder called *peeling algorithm* (Luby et al., 2001). To begin, we consider the contracted graph $\tilde{G}$ in which for each cluster $G^{(\ell)}$, we contract its set of constraint nodes $y_1^{(\ell)}, \ldots, y_{m_\ell}^{(\ell)}$ into a single node $v^{(\ell)}$ (see Figure 2). The contraction graph $\tilde{G}$ is closely related to recursive autoencoders introduced in (Socher et al., 2011). Let us denote the degree distributions of $\widetilde{G}$ (from the edge perspective) by $\widetilde{\lambda}$ and $\widetilde{\rho}$. We say that the node $v^{(\ell)}$ is unsatisfied if it is connected to a noisy pattern node.

The asymptotic performance of our error recovery method, shown in Algorithm 3, is given by the following theorem.

Theorem 3 *Under the assumptions that graph $\widetilde{G}$ grows large and it is chosen randomly with degree distributions given by $\widetilde{\lambda}$ and $\widetilde{\rho}$, Algorithm 3 is successful if $p_e \cdot \widetilde{\lambda}(1 - \widetilde{\rho}(1 - z)) < z$ for $z \in (0, p_e)$.*

The proof is based on the paper (Luby et al., 2001), detailed in the Supplementary Materials.

Algorithm 3 Sequential Peeling Algorithm

Input: $\widetilde{G}, G^{(1)}, G^{(2)}, \ldots, G^{(L)}$.
Output: $x_1, x_2, \ldots, x_n$
1: **while** there is an unsatisfied $v^{(\ell)}$ **do**
2: **for** $\ell = 1 \to L$ **do**
3: If $v^{(\ell)}$ is unsatisfied, apply Algorithm 2 to cluster $G^{(l)}$.
4: If $v^{(\ell)}$ remained unsatisfied, revert the state of pattern neurons connected to $v^{(\ell)}$ to their initial state. Otherwise, keep their current states.
5: **end for**
6: **end while**
7: Declare $x_1, x_2, \ldots, x_n$ if all $v^{(\ell)}$'s are satisfied. Otherwise, declare failure.

The condition given in Theorem 3 can be used to calculate the maximal fraction of errors Algorithm 3 can correct for the given degree distributions. For instance, for the the degree distribution pair $(\widetilde{\lambda}(z) = z^2, \widetilde{\rho}(z) = z^5)$, the threshold is $p_e \approx 0.429$, below which Algorithm 3 corrects all the errors with high probability. Note that the predicted threshold by Theorem 3 is based on the assumption that a cluster can only correct a single error. In practice, as we noted earlier, a cluster can correct more. Hence, the threshold predicted by Theorem 3 is a lower bound on the overall recall performance of our neural network (as the size of the network grows).

6. Pattern Retrieval Capacity

We first note that the number of patterns $\mathcal{C}$ does not have any effect on the learning or recall algorithm except for its obvious influence on the learning time. As long as the patterns come from a subspace, learning Algorithm 1 will yield a matrix W which is orthogonal to all the patterns of the training set. In the recall phase, the proposed denoising algorithms 2 and 3 deal only with $W \cdot e$, where e is the noise vector. In order to show that the pattern retrieval capacity is exponential in n, all we need to demonstrate is that there exists a training set $\mathcal{X}$ with $\mathcal{C}$ patterns of length n for which $\mathcal{C} \propto a^{rn}$, for some $a > 1$ and $0 < r$.

Theorem 4 *Let $\mathcal{X}$ be a $\mathcal{C} \times n$ matrix, formed by $\mathcal{C}$ vectors of length n with entries from the set $\mathcal{S}$. Furthermore, let $k = rn$ for some $0 < r < 1$. Then, there exists a set of vectors for which $\mathcal{C} = a^{rn}$, with $a > 1$, and rank$(\mathcal{X}) = k < n$.*

The proof of this theorem is by construction. The details are outlined in the Supplementary Materials.

7. Simulation Results

We have performed simulations over synthetic and natural databases to investigate the performance of the proposed algorithm and confirm the accuracy of our theoretical analysis.

7.1. Synthetic Dataset

There is a systematic way of generating patterns satisfying a set of linear constraints. More specifically, we generate a matrix $G \in \mathbb{R}^{k \times n}$ of rank $k = r \cdot n$ $(0 < r < 1)$ such that all the entries are non-negative and lie between 0 and $\gamma - 1 > 0$. We construct the patterns of the dataset by setting $x = u \cdot G$, where $u \in \mathbb{R}^k$ is an integer-valued random vector whose entries lie between 0 and $\upsilon - 1 > 0$. We select γ, υ, and G such that all entries of x be less than S. This realization is based on the constructive proof of Theorem 4 and more details can be found in the supplementary materials.

In our simulations, we assume that each pattern neuron is connected to approximately 5 clusters. The number of connections should be neither too small (to ensure information propagation) nor too big (to adhere to the sparsity requirement).

In the learning phase, Algorithm 1 is run in parallel for each cluster which results in learning the constraints. In the recall phase, at each round, a pattern x is sampled uniformly at random from the training set. Then, each of its entries gets corrupted independently with probability p_e. Afterwards, Algorithm 3 is used to denoise the corrupted pattern. We repeat this process many times to calculate the error rate, and compare it to the bound derived in section 5.2.

7.2. Learning Results

The left panels in Figures 3 illustrate the degree distributions of pattern and constraint neurons, respectively, over an ensemble of 5 randomly generated simulation setups. The horizontal axis shows the normalized degree of pattern (resp., constraint) neurons and the vertical axis represents the fraction of neurons with the given normalized degree. The parameters for the learning algorithm are $\alpha_t \propto 0.95/t$, $\beta = 0.75/\alpha_t$ and $\theta_t = 0.05$.

We have executed the learning algorithm for different network sizes and learning parameters. Qualitatively, they all look similar to Figure 3. Moreover, in almost all cases, the learning phase converged within two learning iterations, i.e. by going over the data set only twice.

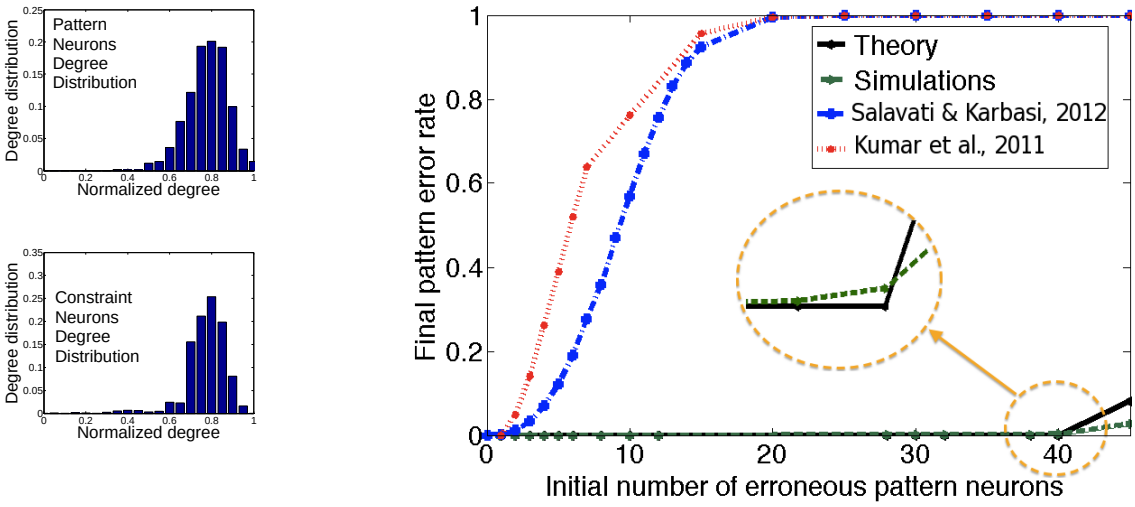

Figure 3: Pattern and constraint neuron degree distributions for $n = 400$, $L = 50$, and on average of 20 constraints per cluster. The learning parameters are $\alpha_t \propto 0.95/t$, $\beta = 0.75/\alpha_t$ and $\theta_t \propto 0.05/t$.

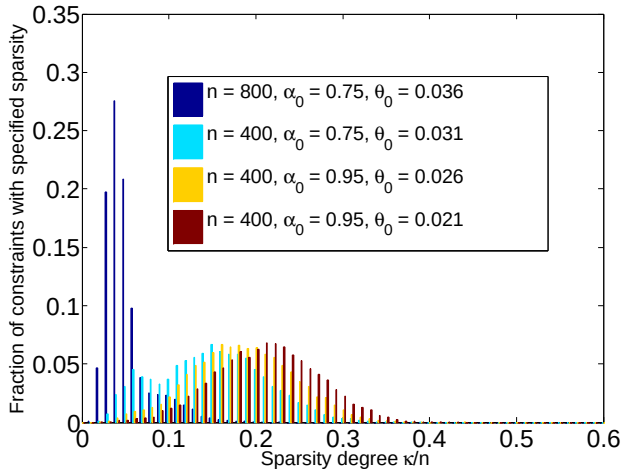

Figure 4: The percentage of variable nodes with the specified sparsity measure.

Effect of θ_t: An interesting issue to investigate is the effect of the hyper-parameter $\theta_t = \theta_0/t$, which is the sparsity threshold. Intuitively, the higher the θ_0 is, the sparser the network will be. Figure 4 confirms this expectation. The figure corresponds to a single cluster with $n = 400$ and $n = 800$ pattern nodes. The horizontal axis of the figure is the sparsity of each weight vector, defined as $\varrho = \kappa/n$, with κ being the number of non-zero elements in the n-dimensional vector. The vertical axis represents the percentage of the weight vectors with the given sparsity measure.

Fig. 4 illustrates two trends: as the sparsity threshold (θ_0) grows, the network becomes sparser. Furthermore, when the network becomes larger, the connections will also become relatively sparser.

7.3. Recall Results

The top panel of Figure 3 illustrates the performance of the recall algorithm. The horizontal and vertical axes represent the number of initial erroneous neurons and the final pattern error rate, respectively. The performance is compared with the theoretical bound derived in section 5.2, as well as the results of the algorithms proposed in (Salavati & Karbasi, 2012) and (Kumar et al., 2011). Note the slight difference between the theoretical and simulation results in the low noise regime (emphasized in the right panel of Figure 3), which is due to the following facts: 1) we stop Algorithm 3 after a limited number of iterations, $t_{\max}$, and 2) the network size is small. The threshold predicted by Theorem 3 becomes accurate as $n \to \infty$.

8. Future Work

We believe that the same method could easily be extend to natural datasets, as they exhibit weak minor components in their correlation matrix. In this case, these datasets could *approximately* be mapped to a subspace which could be then learned by our algorithm. A good case in point is the dataset of natural images belonging to a particular class, which is in fact part of our ongoing research.

Acknowledgments

The authors would like to thank anonymous reviewers for their helpful comments. This work was supported by Grant 228021-ECCSciEng of the European Research Council.

References

Bottou, Leon. Online algorithms and stochastic approximations. In Saad, David (ed.), *Online Learning and Neural Networks*. Cambridge University Press, 1998.

Candès, Emmanuel J. and Tao, Terence. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Inform. Theor.*, 52(12):5406–5425, 2006.

Donoho, D. L. Compressed sensing. *IEEE Trans. Inform. Theor.*, 52(4):1289–1306, 2006.

Gripon, Vincent and Berrou, Claude. Sparse neural networks with large learning diversity. *IEEE Trans. Neur. Netw.*, 22(7):1087–1096, July 2011.

Hopfield, John J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. U.S.A.*, 79(8):2554–2558, 1982.

Jankowski, Stanislaw, Lozowski, Andrzej, and Zurada, Jacek M. Complex-valued multistate neural associative memory. *IEEE Trans. Neural Netw. Learning Syst.*, 7(6):1491–1496, 1996.

Jarrett, Kevin, Kavukcuoglu, Koray, Ranzato, Marc'Aurelio, and LeCun, Yann. What is the best multi-stage architecture for object recognition? In *IEEE Int. Conf. Computer Vision (ICCV)*, pp. 2146–2153, 2009.

Kumar, K. Raj, Salavati, Amir Hesam, and Shokrollahi, Amin. Exponential pattern retrieval capacity with non-binary associative memory. In *IEEE Inf. Theor. workshop (ITW)*, pp. 80–84, Oct 2011.

Le, Quoc V., Ngiam, Jiquan, Chen, Zhenghao, hao Chia, Daniel Jin, Koh, Pang Wei, and Ng, Andrew Y. Tiled convolutional neural networks. In *Proc. Advances in Neur. Inf. Process. Sys. (NIPS)*, pp. 1279–1287, 2010.

Luby, Michael, Mitzenmacher, Michael, Shokrollahi, Amin, and Spielman, Daniel A. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, 2001.

McEliece, Robert J., Posner, Edward C., Rodemich, Eugene R., and Venkatesh, Santosh S. The capacity of the hopfield associative memory. *IEEE Trans. Inf. Theor.*, 33(4):461–482, July 1987.

Muezzinoglu, Mehmet Kerem, Guzelis, Cuneyt, and Zurada, Jacek M. A new design method for the complex-valued multistate hopfield associative memory. *IEEE Trans. Neur. Netw.*, 14(4):891–899, July 2003.

Oja, Erkki and Karhunen, Juha. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Math. Analysis and Applications*, 106:69–84, 1985.

Oja, Erkki and Kohonen, Teuvo. The subspace learning algorithm as a formalism for pattern recognition and neural networks. In *IEEE Int. Conf. Neur. Netw.*, volume 1, pp. 277–284, Jul 1988.

Peretto, P. and Niez, J. J. Long term memory storage capacity of multiconnected neural networks. *Biol. Cybern.*, 54(1):53–64, May 1986.

Salavati, Amir Hesam and Karbasi, Amin. Multi-level error-resilient neural networks. In *Proc. IEEE Int. Symp. Inf. Theor. (ISIT)*, pp. 1064–1068, Jul 2012.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. EMNLP*, pp. 151–161, 2011.

Tanner, R. A recursive approach to low complexity codes. *IEEE Trans. Inf. Theor.*, 27(5):533–547.

Venkatesh, Santosh S. and Psaltis, Demetri. Linear and logarithmic capacities in associative neural networks. *IEEE Trans. Inf. Theor.*, 35(3):558–568, September 1989.

Vincent, Pascal, Larochelle, Hugo, Bengio, Yoshua, and Manzagol, Pierre-Antoine. Extracting and composing robust features with denoising autoencoders. In *Proc. Int. Conf. on Machine Learning (ICML)*, ICML '08, pp. 1096–1103, 2008.

Xu, Lei, Krzyzak, Adam, and Oja, Erkki. Neural nets for dual subspace pattern recognition method. *Int. J. Neural Syst.*, 2(3):169–184, 1991.