Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Associative Memory Using Dictionary Learning and Expander Decoding

Arya Mazumdar
College of Information & Computer Science
University of Massachusetts Amherst
arya@cs.umass.edu

Ankit Singh Rawat*
Research Laboratory of Electronics
Massachusetts Institute of Technology
asrawat@mit.edu

Abstract

An associative memory is a framework of content-addressable memory that stores a collection of message vectors (or a *dataset*) over a neural network while enabling a neurally feasible mechanism to recover any message in the dataset from its noisy version. Designing an associative memory requires addressing two main tasks: 1) *learning phase*: given a dataset, learn a concise representation of the dataset in the form of a graphical model (or a neural network), 2) *recall phase*: given a noisy version of a message vector from the dataset, output the correct message vector via a neurally feasible algorithm over the network learnt during the learning phase. This paper studies the problem of designing a class of neural associative memories which learns a network representation for a large dataset that ensures correction against a large number of adversarial errors during the recall phase. Specifically, the associative memories designed in this paper can store dataset containing $\exp(n)$ n-length message vectors over a network with $O(n)$ nodes and can tolerate $\Omega(\frac{n}{\text{polylog}n})$ adversarial errors. This paper carries out this memory design by mapping the learning phase and recall phase to the tasks of dictionary learning with a square dictionary and iterative error correction in an expander code, respectively.

1 Introduction

Associative memories aim to address a problem that naturally arises in many information processing systems: given a dataset $\mathcal{M}$ which consists of n-length vectors, design a mechanism to concisely store this dataset so that any future query corresponding to a noisy version of one of the vectors in the dataset can be mapped to the correct vector. An associative memory based solution to this problem is broadly required to have two key components: 1) dataset must be stored in the form of a neural network (graph) and 2) the mechanism to map a noisy query to the associated valid vector should be implementable in an iterative neurally feasible manner over the network (a *neurally feasible* algorithm employs only local computations at the nodes of the corresponding network based on the information obtained from their neighboring nodes). The tasks of learning the graph representation from the dataset and mapping erroneous vectors to the associated correct vectors are referred to as *learning phase* and *recall phase*, respectively.

The overarching goal of designing an associative memory that can store a large dataset (ideally containing $\exp(n)$ message vectors using a neural network with $O(n)$ nodes) while ensuring robustness to a large number of errors (ideally $\Omega(n)$ errors) during the recall phase has led to multiple research efforts in the literature. The binary Hopfield networks, as studied in (Hopfield 1982; McEliece et al. 1987), provide one of the earliest designs for the associative memories. Given a dataset containing binary vectors from $\{\pm 1\}^n$, Hopfield networks learn this dataset in the form of an n-node weighted graph by employing Hebbian learning (Hebb 2005), i.e., the weighted adjacency matrix of the graph is defined by summing the outer products of all message vectors in the dataset. However, in their most general form, these networks suffer from small capacity. In (McEliece et al. 1987), McEliece et al. show that these networks can only store $O(\frac{n}{\log n})$ message vectors when these messages correspond to arbitrary n-length binary vectors and the recall phase is required to tolerate linear $\Omega(n)$ random errors. This has motivated the researchers to look at various generalizations of Hopfield networks (see, (Gross and Mezard 1984; Jankowski, Lozowski, and Zurada 1996; McEliece and Posner 1985; Muezzinoglu, Guzelis, and Zurada 2003; Tanaka and Edwards 1980) and references therein). However, these solutions again fail to simultaneously achieve both large capacity and error tolerance.

One remedy to small capacity is to design associative memories with structural assumptions on the dataset. This approach has been considered in (Gripon and Berrou 2011; Hillar and Tran 2014; Karbasi, Salavati, and Shokrollahi 2014; Kumar, Salavati, and Shokrollahi 2011; Mazumdar and Rawat 2015; Salavati and Karbasi 2012). In particular in (Gripon and Berrou 2011), Gripon et al. store a dataset comprising $O(n^2)$ sparse vectors in the form of cliques in a neural network. In (Hillar and Tran 2014), Hillar and Tran design a Hopfield network with n nodes that can store $\sim 2^{\sqrt{2n}}/n^{1/4}$ message vectors and is robust against $n/2$ *random* errors. In (Karbasi, Salavati, and Shokrollahi 2014; Kumar, Salavati, and Shokrollahi 2011; Salavati and Karbasi 2012; Mazumdar and Rawat 2015), the message vectors that need to be stored are assumed to consti-

*This work was done when the author was with the Computer Science Department, Carnegie Mellon University, PA, USA.

tute a subspace. In (Karbasi, Salavati, and Shokrollahi 2014; Kumar, Salavati, and Shokrollahi 2011; Salavati and Karbasi 2012), the task is to learn a bipartite factor graph of the linear constraints satisfied by the dataset subspace. The error correction during recall phase is then performed by running a belief propagation algorithm (Richardson and Urbanke 2008) over the bipartite graph. In (Karbasi, Salavati, and Shokrollahi 2014), Karbasi et al. work with a model where the message vectors in the dataset have overlapping sets of coordinates so that shortened vectors obtained by restricting the original message vectors to each of these overlapping sets belong to a subspace. Under this model, they design associative memories that can store exponential number (in n) of message vectors while correcting linear number (in n) of random errors during the recall phase.

The results in (Karbasi, Salavati, and Shokrollahi 2014) hinge on the fact that the learning phase of their memory design recovers a bipartite graph which has certain desirable structural properties that are required for belief propagation type decoders to converge. However, no guarantee of recovering such a bipartite graph during the learning phase is provided in (Karbasi, Salavati, and Shokrollahi 2014) even when we assume the subspace associated with the dataset has one such graphical representation to begin with. Recognizing the requirement of learning correct bipartite graph during the learning phase, we explore a sparse recovery based approach to design associative memories with the subspace dataset model in (Mazumdar and Rawat 2015). This approach assumes that the dataset belongs to a subspace whose orthogonal subspace has null space property, a sufficient condition for sparse signal recovery. This allows one to learn any basis for the orthogonal subspace during the learning phase and then recast the recall phase as a sparse recovery problem (Candes and Tao 2006; Donoho 2006). The approach in (Mazumdar and Rawat 2015) also allows for the strong error model containing *adversarial* errors. Specifically, (Mazumdar and Rawat 2015) considers two candidate signal models which contain n-length message vectors and utilize $O(n)$ sized neural networks to store the signals. The two models have the datasets of sizes $\exp(n^{3/4})$ and $\exp(r)$ with $1 \leq r \leq n$, respectively. Furthermore, the designed associative memories based on these two signal models respectively allow for recovery from $\Omega(n^{1/4})$ and $\Omega\left(\frac{n-r}{\log^6 n}\right)$ adversarial errors in a neurally feasible manner.

In this paper, we also follow the subspace model as in (Karbasi, Salavati, and Shokrollahi 2014; Kumar, Salavati, and Shokrollahi 2011; Salavati and Karbasi 2012; Mazumdar and Rawat 2015). We assume the dataset to form a subspace which is defined by *sparse linear constraints*. The model of sparse linear constraints are quite natural and less restrictive than the previous models of works such as (Mazumdar and Rawat 2015). Note that this signal model is similar to the model explored in Karbasi et al. (Karbasi, Salavati, and Shokrollahi 2014). However, our approach and contributions differ from (Karbasi, Salavati, and Shokrollahi 2014), as we ensure that the learning phase *provably* generates the correct bipartite graph which can guarantee the error correction from a large number of errors using an iterative algorithm during the recall phase. We also note that similar to (Mazumdar and Rawat 2015) we work with the stronger error model involving adversarial errors, but our scheme is superior to that of (Mazumdar and Rawat 2015) in terms of storage capacity (see, Theorems 1, 3) and number of correctable adversarial errors (improvement by poly-log factors, see, Theorems 1, 2, 3). We want to point out that the main technical challenge in associative memory is not to individually design the learning or recall phases, but to interface them in a way that is consistent with the operations of both phases, and to give an end-to-end performance guarantee.

Here, we note that the problem of designing an associative memory is closely related to the well studied nearest neighbor search (NNS) problem and its relaxation approximate nearest neighbor search (A-NNS) problem (Indyk and Motwani 1998; Andoni and Indyk 2008; Samet 2005; Wang et al. 2014). The solutions to the A-NNS problem enable one to store a dataset in such a manner that noisy versions of the vectors in a dataset (with bounded noise) can be mapped to the correct vectors. Additionally, the A-NNS solutions do not put assumptions on the dataset. However, this comes at the cost of removing the requirement of having a fast iterative or neurally feasible recall phase. Furthermore, the A-NNS solutions, especially based on locally sensitive hashing (Indyk and Motwani 1998; Har-Peled, Indyk, and Motwani 2012) have large space complexity, i.e., polynomial in size of dataset. We note that the A-NNS solutions are very much aligned to the vector (image) retrieval task (Jégou, Douze, and Schmid 2011; Yu et al. 2015; Ferro, Gripon, and Jiang 2016) which need not have a neurally feasible retrieval algorithm.

The rest of the paper is organized as follows. In Sec. 2, we define the dataset model considered in this paper and present the main results of this paper along with key techniques and ideas involved in establishing those results. Sec. 3 is dedicated to the proof of the main theorem. In Sec. 3.1, we describe the learning phase of the associative memory design results along with the relevant technical details. In Sec. 3.2, we present an iterative error correction algorithm which is employed during the recall phase of the designed associative memory. This analysis of the algorithm relies on the expansion properties of the bipartite graph which defines the dataset and is learnt during the learning phase. We conclude the paper with some comments on performance in Sec. 5.

2 Main results and techniques

2.1 Model for datasets

We focus on the associative memories based on the operations on $\mathbb{R}$, the set of real numbers. In our first model, we consider the message patterns to be vectors over $\mathbb{R}$. In the second model we comment on neural associative memories storing binary message patterns that are obtained by our approach.

Dataset over real numbers: the sparse-sub-Gaussian model We assume the message set to form a linear subspace defined by sparse linear constraints over $\mathbb{R}$. Let $\mathcal{M} \subseteq \mathbb{R}^n$ denote the set of message vectors (signals) that need to be stored on the associative memory. Let B be an $m \times n$ matrix comprising the linear constraints that define the message set

$\mathcal{M}$. In particular, we have

$$B\boldsymbol{x} = 0 \quad \forall\, \boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{M}. \tag{1}$$

In order to fully specify the message set $\mathcal{M}$, we still need to provide a stochastic model for the matrix B. Towards this, we consider a random ensemble of sparse matrices. For each $j \in [n] := \{1, 2, \ldots, n\}$, we consider the following experiment. We pick d element uniformly at random with replacement from the set $[m]$. Let $\mathcal{N}_j$ denote the set comprising these randomly picked elements. For $1 \leq i \leq m,\ 1 \leq j \leq n$, we define

$$\xi_{i,j} = \begin{cases} 1 & \text{if } i \in \mathcal{N}_j \subset [m] \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Let $\left\{R_{i,j}\right\}_{1 \leq i \leq m,\ 1 \leq j \leq n}$ be a collection of independent and identically distributed (i.i.d.) sub-Gaussian random variables. Given the random variables, $\left\{\xi_{i,j}, R_{i,j}\right\}_{1 \leq i \leq m,\ 1 \leq j \leq n}$, we assume that the (i,j)-th entry of the matrix B is defined as

$$B_{i,j} = \xi_{i,j} R_{i,j} \in \mathbb{R} \quad \text{for } 1 \leq i \leq m,\ 1 \leq j \leq n. \tag{3}$$

Through out this paper, we refer to this model for the dataset to be stored on a neural associative memory as *sparse-sub-Gaussian model*. We work with various values of d which we specify while stating different parameters that we obtain for the designed associative memories in Sec. 2.2.

This model is a quite natural random model of bipartite graphs that allow for multi-edges. Indeed, consider a bipartite graph with disjoint sets of vertices $[n]$ (variable nodes) and $[m]$ (check nodes). There are d edges out of each variable node, being incident on uniformly and independently chosen vertices from the check nodes.

Remark 1. *The requirement on $R_{i,j}$ is quite generic as it allows for many distributions. For example, we can assume that $R_{i,j}$ belongs to a finite set of integers $\{-L, -L+1, \ldots, -1, 1, \ldots, L-1, L\}$. Similarly, in another setup, $R_{i,j}$ can be assumed to be a Gaussian random variable.*

Binary dataset Our model of binary dataset is same as above except for the fact that 1) $\mathcal{M} \subseteq \{+1, -1\}^n$, and 2) $R_{i,j}$ is uniform over $\{+1, -1\}$ in (3). The condition of (1) must be satisfied for any $\boldsymbol{x} \in \mathcal{M}$.

2.2 Our main results

We establish that, for a dataset $\mathcal{M}$ corresponding to the null-space defined by the matrix B, the said matrix B can be exactly recovered from the dataset in polynomial time. Recall that there can be many sets of basis-vectors for the null-space of $\mathcal{M}$. Still, we claim that it is possible to accurately recover the matrix B that has been generated by the sparse-sub-Gaussian model described above.

It is essential for us that we recover the matrix B exactly. Being generated by the random model defined above, B exhibits certain graph expansion property that is necessary for our recall phase to be successful. This matrix B enables the error correction during the recall phase with the help of a simple iterative (neurally feasible algorithm). We summarize the parameters achieved by such memory as follows.

Theorem 1. *Suppose that $c, c', c'' > 0$ are three constants. Let n be a large enough integer and $m = c\frac{n}{\log n}$. Assume that B is an $m \times n$ matrix generated from the sparse-sub-Gaussian model described in Sec. 2.1 with $c' \leq d \leq c'' \log n$, and $\mathcal{M} = \{\boldsymbol{x} \in \mathbb{R}^n :\ B\boldsymbol{x} = 0\}$. Then, with high probability (w.h.p.) $\mathcal{M}$ is an $n - m = n(1 - c/\log n)$ dimensional subspace that can be stored in a neural network (learned in poly-time in the learning phase) while allowing for correct recovery from $\Omega(\frac{n}{d^2 \log^2 n})$ adversarial errors during the recall phase with a neurally feasible algorithm.*

The proof of this theorem has been provided in Sec. 3. This result is obtained by utilizing a novel connection between recovering the matrix B defining the underlying dataset $\mathcal{M}$ and the dictionary learning problem with a square dictionary as studied in (Spielman, Wang, and Wright 2012; Adamczak 2016; Blasiok and Nelson 2016). Given access to the dataset $\mathcal{M}$, we can easily find a basis for the null-space of $\mathcal{M}$ containing $m = n - \dim(\mathcal{M})$ n-length vectors. Let A denote the $m \times n$ matrix which has the m vectors in this basis as its rows. Note that the row vectors of B also span the subspace orthogonal to the dataset $\mathcal{M}$. Moreover, w.h.p., B is a full rank matrix. This implies that the following relationship holds w.h.p.,

$$A = DB, \tag{4}$$

where D is an invertible $m \times m$ matrix. Note that recovering the matrix B from A is now equivalent to dictionary learning problem (Olshausen and Field 1997) where n columns of A and B corresponds to n observations and the associated coefficients, respectively. Furthermore the matrix D corresponds to a square dictionary (Spielman, Wang, and Wright 2012).

As for the recall phase, we rely on the observations (as shown in Sec. 3.2) that w.h.p. the bipartite graph associated with the sparse random matrix B is an expander graph. Assume that we are given a noisy version $\boldsymbol{y}$ of a valid message vector $\boldsymbol{x} \in \mathcal{M}$ such that we have

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e} \tag{5}$$

where $\boldsymbol{e}$ denotes the error vector. Recovering $\boldsymbol{x}$ from the observation $\boldsymbol{y}$ can be cast as a sparse recovery problem of recovering $\boldsymbol{e}$ from $\boldsymbol{z} = B\boldsymbol{y} = B(\boldsymbol{x} + \boldsymbol{e}) = B\boldsymbol{e}$. If the bipartite graphs associated with B is an expander graph (which holds w.h.p.), we can solve this sparse recovery problem by an efficient and iterative algorithm (Jafarpour et al. 2009) which is motivated by the decoding algorithm of expander codes (Sipser and Spielman 1996) in coding theory literature.

Due to the sample complexity requirements for efficient square-dictionary learning algorithms (Spielman, Wang, and Wright 2012; Adamczak 2016; Blasiok and Nelson 2016), the above model allows us to store datasets that satisfy at most $O\big(\frac{n}{\log n}\big)$ linear constraints. However if we allow for a learning-phase that takes quasi-polynomial time, then it is possible to store restricted datasets that satisfy $m = \Theta(n)$ sparse-linear constraints. We summarize the result below.

Theorem 2. *Let n be a large enough integer and $m = cn$ for a some constant $c < 1/200$. For a large enough constant $C > 0$, let B be an $m \times n$ matrix generated from the sparse-sub-Gaussian model described in Sec. 2.1 with $d = C \log n$*

and $\mathcal{M} = \{\boldsymbol{x} \in \mathbb{R}^n : B\boldsymbol{x} = 0\}$. Then w.h.p., $\mathcal{M}$ is an $n - m = n(1 - c)$ dimensional subspace that can be stored in a neural network (learned in quasi-polynomial-time in the learning phase) while allowing for error correction from $\Omega(\frac{n}{\log^2 n})$ adversarial errors during the recall phase with a neurally feasible algorithm.

While in terms of storage capacity this theorem is inferior to that of Theorem 1, it may represent some datasets better, and has better error correction capability. While the recall phase of this algorithm works same as above, for the learning phase we can no longer rely on the dictionary-learning algorithms. Instead we do an exhaustive search over all possible sparse vectors to find out a sparse basis for the null-space of $\mathcal{M}$ which end up taking a quasi-polynomial time, if we choose parameters suitable for the recall phase. We here crucially use the fact that for $m = cn$ and $d = C \log n$ such a sparse basis is unique, which can be obtained from the results of (Spielman, Wang, and Wright 2012). The proof of the recall phase for this theorem remains same as that of Theorem 1.

Finally, while both Theorems 1 and 2 have their counterparts when storing binary vectors, we present only one result for brevity. A sketch of the proof of the following theorem has been given in Sec. 4.

Theorem 3 (Binary dataset)**.** *Suppose that $c, c', c'' > 0$ are three constants. Let n be a large enough integer such that $m = c\frac{n}{\log n}$. Assume that B is an $m \times n$ matrix generated from the binary dataset model described in Sec. 2.1 with $c' \leq d \leq c'' \log n$ and $\mathcal{M} = \{\boldsymbol{x} \in \{\pm 1\}^n : B\boldsymbol{x} = 0\}$. Then w.h.p., $|\mathcal{M}| = \exp(n - \alpha n \log(d \log n)/\log n)$ for a constant α and $\mathcal{M}$ can be stored in a neural network (learned in polynomial-time in the learning phase) while allowing for error correction from $O(\frac{n}{d^2 \log^2 n})$ adversarial errors during the recall phase with a neurally feasible algorithm.*

3 Proof of Theorem 1

3.1 Learning phase of associative memory design

As discussed in the previous section, under the dataset model considered in this paper, the learning phase of the associative memory design can be mapped to the problem of dictionary learning with a square dictionary. The very same dictionary learning problem with slightly different random model for the coefficient vector has been studied in (Spielman, Wang, and Wright 2012; Adamczak 2016; Blasiok and Nelson 2016). (We refer the reader to a longer version of this paper (Mazumdar and Rawat 2016) for a description of this line of work.) We then utilize the dictionary learning algorithm used in (Adamczak 2016) to exactly learn the matrix B which define our dataset and comment on the modifications required in the analysis of Adamczak (Adamczak 2016) to obtain guarantees on the performance of this algorithm.

Exact recovery of the matrix B Our learning phase constitutes learning the matrix B exactly from the dataset $\mathcal{M}$. Utilizing the dictionary learning algorithm from (Adamczak 2016), we design the learning phase for an associative memory storing the message set described in Sec. 2.1. The learning phase consists of the following two steps.

1. Given the message vectors from the dataset $\mathcal{M}$, first construct a basis for the subspace orthogonal to the dataset subspace $\mathcal{M} = \{\boldsymbol{x} : B\boldsymbol{x} = 0\} \subset \mathbb{R}^n$ with $\dim(\mathcal{M}) = n - m$.
2. Let $A \in \mathbb{R}^{m \times n}$ denote the basis obtained in the previous step. Since w.h.p. B is a full-rank matrix, we have
$$A = DB,$$
where $D \in \mathbb{R}^{m \times m}$ is a non-singular matrix. Now employ the modified ERSpUD dictionary learning algorithm (Adamczak 2016) with the matrix A as its input. Note that the algorithm outputs candidates for the matrices D and B. The method of this square-dictionary learning and the algorithm are summarized in the longer version of this paper (Mazumdar and Rawat 2016).

Next, we show that the proposed learning phase w.h.p. exactly recovers the matrix B. Note that the sparse-sub-Gaussian model used to generate B (cf. Sec. 2.1) slightly differs from the Bernoulli-sub-Gaussian model studied in (Spielman, Wang, and Wright 2012; Adamczak 2016) (cf. (Mazumdar and Rawat 2016, Appendix A)). In particular, for every $j \in [n]$, the distribution of the random variables $\{\xi_{i,j} : i \in [m]\}$ and $\{\eta_{i,j} : i \in [m]\}$ is different[1]. However, this difference is not very crucial for the success of the learning algorithm as we still have independence among the random variables $\xi_{i,j}$s which are indexed by different values of $j \in [n]$. We formalize the exact recovery guarantees for the matrix B in the following result.

Theorem 4. *Let $B \in \mathbb{R}^{m \times n}$ be a matrix generated by the sparse-sub-Gaussian model (cf. Sec. 2.1) and $\mathcal{M}$ be the associated dataset, i.e., $\mathcal{M} = \{\boldsymbol{x} : B\boldsymbol{x} = 0\}$. Then there exists a constant $c > 0$ such that whenever we have $n \geq cm \log m$ the two step learning phase of the associative memory as described above exactly recovers the linear constraints in B with probability at least $1 - 1/n$.*

We refer the reader to (Mazumdar and Rawat 2016) for the proof of Theorem 4.

3.2 Recall phase of associative memory design

In this section we present an iterative algorithm which recovers the correct message vector among the dataset $\mathcal{M}$ from its noisy version. The noisy observation is assumed to be corrupted at adversarially chosen coordinates. The correctness of the iterative algorithm relies on the observation that the bipartite graph associated with the matrix B which defines our dataset $\mathcal{M}$ is a good expander graph. We first formalize this expansion property in the following result. We then present the iterative algorithm and show that it can provably tolerate $\Omega\big(\frac{n}{\text{polylog}n}\big)$ adversarial errors.

[1]We focus on the sparse-sub-Gaussian model as opposed to the Bernoulli-sub-Gaussian model as the bipartite graph associated with the matrix B generated by the sparse-sub-Gaussian model is a good expander w.h.p. We utilize this fact while designing the recall phase for the proposed associative memory in Sec. 3.2.

Expansion property of the bipartite graph defined by B Let $\mathcal{G}_B = (\mathcal{L} = [n], \mathcal{R} = [m], \mathcal{E}_B)$ be a bipartite graph where $\mathcal{L}$ and $\mathcal{R}$ denote the index sets of left and right vertices, respectively. The matrix B which defines our dataset $\mathcal{M}$ gives the $m \times n$ adjacency matrix of the graph $\mathcal{G}$, i.e., for $\ell \in \mathcal{L}$ and $r \in \mathcal{R}$, we have an edge $(\ell, r) \in \mathcal{E}_B$ iff $B_{r,\ell} \neq 0$. More specifically, the weight of the edge $(\ell, r) \in \mathcal{E}_B$ is $w_{\ell,r} = B_{r,\ell}$. It follows from the sparse-sub-Gaussian model (cf. Sec. 2.1) which generates the random matrix B that every vertex in $\mathcal{L}$ has degree d and each of the d neighbors for a vertex in $\mathcal{L}$ are chosen uniformly at random from the set of right vertices $\mathcal{R}$ with replacement. The following result states that expansion properties that hold for such a graph with high probability.

Proposition 1. *Assume that $\epsilon > 0$ and $d = O(\frac{n}{m \log n})$. Let $\mathcal{G} = (\mathcal{L}, \mathcal{R}, \mathcal{E})$ be a random d-left regular graph where each of the d neighbors for a left vertex are chosen uniformly at random from the set of right vertices with replacement. Then, for a large enough n, w.h.p., $\mathcal{G}$ is an $\left(\frac{m^2}{d^2 n}, (1-\epsilon)d\right)$-expander graph, where a bipartite graph is (t, l)-expander, if for every $\mathcal{S} \subseteq \mathcal{L}$ such that $|\mathcal{S}| \leq t$, we have $|\mathcal{N}(\mathcal{S})| \geq l|\mathcal{S}|$. Here, $\mathcal{N}(\mathcal{S}) \subseteq \mathcal{R}$ denotes the vertices in $\mathcal{R}$ that are neighbors of vertices in $\mathcal{S}$.*

Proof. Let's consider a set $\mathcal{S} \subseteq \mathcal{L}$ such that $|\mathcal{S}| = s \leq \frac{m^2}{d^2 n}$. Let $\mathcal{T} \subseteq \mathcal{R}$ be a set of right vertices such that $|\mathcal{T}| < (1-\epsilon)ds$. The probability that $\mathcal{N}(\mathcal{S}) \subseteq \mathcal{T}$ is upper bounded by $\left(\frac{(1-\epsilon)ds}{m}\right)^{ds}$. Now, taking the union bound over all the sets $\mathcal{S} \subseteq \mathcal{L}$ such that $|\mathcal{S}| = s$ and the sets $\mathcal{T} \subseteq \mathcal{R}$ such that $|\mathcal{T}| < (1-\epsilon)ds$, the probability P_s that the the graph $\mathcal{G}$ has a non-expanding set of size s, is upper bounded as follows.

$$\begin{aligned} P_s &\leq \binom{n}{s}\binom{m}{(1-\epsilon)ds}\left((1-\epsilon)ds/m\right)^{ds} \\ &\leq e^{s+(1-\epsilon)ds}\left(n/s\right)^{s}\left((1-\epsilon)ds/m\right)^{\epsilon ds}. \end{aligned} \tag{6}$$

We can rewrite (6) as,

$$P_s \leq e^{s+(1-\epsilon)ds}\left(dn/m\right)^{s}\left(ds/m\right)^{\epsilon ds - s}. \tag{7}$$

Now, using our assumption that $s \leq \frac{m^2}{d^2 n}$, we obtain that

$$P_s \leq e^{s+(1-\epsilon)ds}\left(m/dn\right)^{\epsilon ds - 2s}. \tag{8}$$

Using union bound, we have that $\mathcal{G}$ is not an $\left(\frac{m^2}{d^2 n}, (1-\epsilon)d\right)$-expander with probability at most

$$\sum_{s=1}^{\frac{m^2}{d^2 n}} P_s \leq \frac{m^2}{d^2 n} e^{s+(1-\epsilon)ds}\left(\frac{m}{dn}\right)^{\epsilon ds - 2s}. \tag{9}$$

Now, for large enough n, the R.H.S. of (9) vanishes as we have $\frac{m}{dn} = O(\frac{1}{\log n})$ □

Iterative decoding algorithm Remember that during the recall phase we are given an n-length observation vector $\boldsymbol{y}$ which is noisy version of one of the message vectors from the dataset $\mathcal{M}$, i.e.,

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}, \text{ for some } \boldsymbol{x} \in \mathcal{M}. \tag{10}$$

Expander decoding algorithm

Input: The vector $\boldsymbol{z} = B\boldsymbol{e}$ and the matrix B.
1: Define $\mathcal{N}_j := \{i \in [m] \ : \ B_{i,j} \neq 0\} \ \forall \ j \in [n]$.
2: Initialize $\widehat{\boldsymbol{e}} = 0$.
3: **if** $\boldsymbol{z} = B\widehat{\boldsymbol{e}}$ **then**
4: End the decoding and output $\widehat{\boldsymbol{e}}$.
5: **else**
6: Find an index $j \in [n]$ such that the multiset $\{\frac{g_i}{B_{i,j}}\}_{i \in \mathcal{N}_j}$ has at least $(1-2\epsilon)d$ identical elements, say δ. Here, g_i is the gap (cf. (12)) of the constraint defined by the ith row of B.
7: Set $\widehat{e}_j \leftarrow \widehat{e}_j + \delta$ and go to 2.
8: **end if**

Figure 1: Recovery algorithm for sparse vector from expander graphs based measurement matrix (Jafarpour et al. 2009).

Assuming that we have exactly learnt the $m \times n$ matrix B during the learning phase of the associative memory (as described in Sec. 3.1), we obtain an m-length vector as follows.

$$\boldsymbol{z} = B\boldsymbol{y} = B(\boldsymbol{x} + \boldsymbol{e}) = B\boldsymbol{e}, \tag{11}$$

where the last equality follows as we have $\boldsymbol{x} \in \mathcal{M} = \{\boldsymbol{x} \in \mathbb{R}^n \ : \ B\boldsymbol{x} = 0\}$. Note that we have reduced the problem of recovery of the correct message vector $\boldsymbol{x}$ from $\boldsymbol{y}$ to the task of recovering $\boldsymbol{e}$ from $\boldsymbol{z}$. Assuming that the error vector $\boldsymbol{e}$ satisfies certain sparsity constraint, the latter problem is exactly the problem of recovering the sparse vector $\boldsymbol{e}$ from its linear measurements via the measurement matrix B. As shown in Proposition 1, w.h.p., the matrix B corresponds to the adjacency matrix of an expander graph. In (Jafarpour et al. 2009), Jafarpour et al. have adapted the iterative error correction algorithm for expander codes from (Sipser and Spielman 1996) to the problem of sparse recovery problem when the measurement matrix corresponds the adjacency matrix of a good expander graph. Here we propose to employ this iterative algorithm to recover $\mathbf{e}$ from $\boldsymbol{z}$. The algorithm requires calculation of *gap* for each of the linear constraints defined by the matrix B (or rows of the matrix B) which we formally define below.

Definition 1. Let $\boldsymbol{e}$ be an error vector and $\boldsymbol{z} = B\boldsymbol{e}$. Given an estimate $\widehat{\boldsymbol{e}}$ for $\boldsymbol{e}$, for each linear constraint indexed by $i \in [m]$, we define a gap g_i as follows.

$$g_i = z_i - \sum_{j=1}^{n} B_{i,j}\widehat{e}_j. \tag{12}$$

We describe the algorithm in Fig. 1 and present the theoretical guarantees for the performance of the algorithm from (Jafarpour et al. 2009) as follows.

Proposition 2 ((Jafarpour et al. 2009))**.** *Let B be an $m \times n$ matrix which is the adjacency matrix for a $(2k, (1-\epsilon)d)$ expander bipartite graph with $\epsilon \leq \frac{1}{4}$. Then, given the measurement vector $\boldsymbol{z} = B\boldsymbol{e}$ for any k-sparse vector $\boldsymbol{e}$, the expander decoding algorithm (cf. Fig. 1) successfully recovers $\mathbf{e}$ in at most $2k$ iterations.*

We now employ Proposition 2 to characterize the error correction performance of the designed associative memories during the recall phase.

Theorem 5. *Let B be the $m \times n$ matrix generated by the sparse-sub-Gaussian model described in Sec. 2.1 and $\mathcal{M}$ denote the dataset associated with the matrix B. Then, with probability at least $1 - o(1)$, the recall phase based on the iterative decoding algorithm described in Fig. 1 can correct at least $\frac{m^2}{2d^2n}$ adversarial errors.*

Proof. It follows from Proposition 1 that with probability at least $1 - o(1)$, the matrix B corresponds to the adjacency matrix of an $\left(\frac{m^2}{d^2n}, (1-\epsilon)d\right)$-expander graph. Combining the expansion parameters for this expander graph with the result in Proposition 2, we obtain that the iterative decoding algorithm (cf. Fig. 1) can recover the error vector $\boldsymbol{e}$ from $\boldsymbol{z} = B\boldsymbol{e}$ as long as $\boldsymbol{e}$ has at most $\frac{m^2}{2d^2n}$ non-zero coordinates. Given $\boldsymbol{y}$ and $\boldsymbol{e}$, it is straightforward to obtain the correct message vector as $\boldsymbol{x} = \boldsymbol{y} - \boldsymbol{e}$. This completes the proof. □

4 Proof sketch of Theorem 3: Associative memory storing binary vectors

Since the graph defined by B is still an expander (with edge weights $\{+1, -1\}$), for the recall phase we rely on the same expander decoding algorithm. We just want to guarantee that $|\mathcal{M}| = |\{\boldsymbol{x} \in \{\pm 1\}^n : B\boldsymbol{x} = 0\}|$ is of size about $\exp(n - \alpha n \log(d \log n)/\log n)$ w.h.p. The algorithm to learn B is same as that of Theorem 1.

Instead of the random model that we have considered in Sec. 2.1, consider a random matrix $B \in \{+1, 0, -1\}^{m \times n}$ whose each row has independently and uniformly chosen d' nonzero ($\{+1, -1\}$) values. This model allows us to come up with a straight-forward analysis of number of binary vectors in the null-space, while the original model gives the same estimate but with significantly lengthier analysis, that we omit for the interest of space. Note that $d' \sim d\frac{n}{m}$ w.h.p. Now for a randomly and uniformly chosen ± 1 vector $\boldsymbol{y}$ of length n, and for some constant $c' > 0$,

$$\mathbb{P}\{B\boldsymbol{y} = 0\} = \left(\binom{d'}{\frac{d'}{2}}/2^{d'}\right)^m \geq \left(\frac{1}{c'd'}\right)^{m/2}.$$

This means $\mathbb{E}\big[|\mathcal{M}|\big] \geq 2^n \cdot \left(1/(c'd')\right)^{m/2} = 2^{n - \frac{m}{2}\log(c'd')}$. Substituting, $m = c\frac{n}{\log n}$, we get the promised size of $\mathcal{M}$.

5 Simulation results

Though our main contribution is theoretical, in this section we evaluate the proposed associative memory on synthetic dataset to verify if our methods works. Only a representative figure is presented here (Fig. 2). We consider three sets of system parameters (m, n, d) for the dataset to be stored. For each set of parameters, we first generate an $m \times n$ random matrix B according to the sparse-sub-Gaussian model (cf. Sec. 2.1). Each non-zero entry of the matrix B is drawn uniformly at random from the set $\{\pm 1, \pm 2, \pm 3\}$. We then generate multiple message vectors which belong to the subspace orthogonal to all the rows of the matrix B and provide

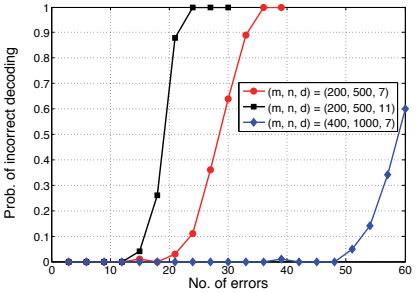

Figure 2: Performance of recall phase for different sets of system parameters.

the learning phase with these vectors. Given these vectors we employ the dictionary learning based approach described in Sec. 3.1 to obtain an estimate $\widehat{B}$ for the matrix B. As guaranteed by Theorem 4, in our simulations, $\widehat{B}$ contains all the rows of the original matrix B (however, in a different order). For all three sets of parameters under consideration, we then utilize the estimate $\widehat{B}$ to evaluate the performance of the expander decoding based recall phase (cf. Sec. 3.2). For a fixed number E of errors, we generate 100 error vectors $\mathbf{e} \in \mathbb{R}^n$ with the number of non-zero entries in each error vector equal to E. The non-zero entries in these vectors are uniformly generated from the set $\{\pm 1, \ldots, \pm 4\}$. The positions of the non-zeros entries in each of these vectors are chosen according to a uniform random permutation on the set $[n]$. The performance of the recall algorithm in our simulations is illustrated in Fig. 2 where we plot the fraction of incorrectly recovered error vectors as we increase the number of errors. As expected from Theorem 5, increasing d while keeping m and n fixed degrades the performance of the recall phase. On the other hand, increasing m while keeping d and the ratio $\frac{m}{n}$ fixed improves the performance of the recall phase.

Concluding remarks. While we use dictionary learning as a tool in the learning phase, the model of our datasets are subspace models. A large number of datasets on the other hand are also modeled by the *sparse dictionary* model (or union of subspaces). It is of interest to design associative memories, where the datasets are modeled as such. One other possible direction of future research would be to consider a subspace model with a mixture of sparse and dense constraints, which potentially will be inclusive of larger classes of real datasets. For such datasets, under suitable assumption on the generative model, one can potentially employ the techniques of recovering planted sparse vectors in a subspace spanned by dense random sub-Gaussian vectors (Demanet and Hand 2014; Qu, Sun, and Wright 2014) and utilize the recovered sparse constraints to design an iterative recall phase similar to the one presented in this paper. As in the case of (Karbasi, Salavati, and Shokrollahi 2014), the networks (graphs) appearing in our associative memory design share some similarities with the neural networks used for classification tasks. It is an interesting problem to further explore such connections.

References

Adamczak, R. 2016. A note on the sample complexity of the er-spud algorithm by spielman, wang and wright for exact recovery of sparsely used dictionaries. *CoRR* abs/1601.0204.

Andoni, A., and Indyk, P. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 51(1):117–122.

Blasiok, J., and Nelson, J. 2016. An improved analysis of the er-spud dictionary learning algorithm. *CoRR* abs/1602.05719.

Candes, E. J., and Tao, T. 2006. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. on Inf. Theory* 52(12):5406–5425.

Demanet, L., and Hand, P. 2014. Scaling law for recovering the sparsest element in a subspace. *Information and Inference*.

Donoho, D. L. 2006. Compressed sensing. *IEEE Trans. on Inf. Theory* 52(4):1289–1306.

Ferro, D.; Gripon, V.; and Jiang, X. 2016. Nearest neighbour search using binary neural networks. In *Proceedings of IJCNN*.

Gripon, V., and Berrou, C. 2011. Sparse neural networks with large learning diversity. *IEEE Transactions on Neural Networks* 22(7):1087–1096.

Gross, D. J., and Mezard, M. 1984. The simplest spin glass. *Nuclear Physics B* 240(4):431 – 452.

Har-Peled, S.; Indyk, P.; and Motwani, R. 2012. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of computing* 8(1):321–350.

Hebb, D. O. 2005. *The organization of behavior: A neuropsychological theory*. Psychology Press.

Hillar, C., and Tran, N. M. 2014. Robust exponential memory in hopfield networks. *CoRR* abs/1411.4625.

Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* 79(8):2554–2558.

Indyk, P., and Motwani, R. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, 604–613. New York, NY, USA: ACM.

Jafarpour, S.; Xu, W.; Hassibi, B.; and Calderbank, R. 2009. Efficient and robust compressed sensing using optimized expander graphs. *IEEE Transactions on Information Theory* 55(9):4299–4308.

Jankowski, S.; Lozowski, A.; and Zurada, J. M. 1996. Complex-valued multistate neural associative memory. *IEEE Transactions on Neural Networks* 7(6):1491–1496.

Jégou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33(1):117–128.

Karbasi, A.; Salavati, A. H.; and Shokrollahi, A. 2014. Convolutional neural associative memories: Massive capacity with noise tolerance. *CoRR* abs/1407.6513.

Kumar, K. R.; Salavati, A. H.; and Shokrollahi, A. 2011. Exponential pattern retrieval capacity with non-binary associative memory. In *2011 IEEE Information Theory Workshop (ITW)*, 80–84.

Mazumdar, A., and Rawat, A. S. 2015. Associative memory via a sparse recovery model. In *Advances in Neural Information Processing Systems (NIPS)*. 2683–2691.

Mazumdar, A., and Rawat, A. S. 2016. Associative memory using dictionary learning and expander decoding. *CoRR* abs/1611.09621.

McEliece, R. J., and Posner, E. C. 1985. The number of stable points of an infinite-range spin glass memory. *Telecommunications and Data Acquisition Progress Report* 83:209–215.

McEliece, R. J.; Posner, E. C.; Rodemich, E. R.; and Venkatesh, S. S. 1987. The capacity of the hopfield associative memory. *IEEE Transactions on Information Theory* 33(4):461–482.

Muezzinoglu, M. K.; Guzelis, C.; and Zurada, J. M. 2003. A new design method for the complex-valued multistate hopfield associative memory. *IEEE Transactions on Neural Networks* 14(4):891–899.

Olshausen, B. A., and Field, D. J. 1997. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research* 37(23):3311 – 3325.

Qu, Q.; Sun, J.; and Wright, J. 2014. Finding a sparse vector in a subspace: linear sparsity using alternating directions. *CoRR* abs/1412.4659.

Richardson, T., and Urbanke, R. 2008. *Modern Coding Theory*. New York, NY, USA: Cambridge University Press.

Salavati, A. H., and Karbasi, A. 2012. Multi-level error-resilient neural networks. In *2012 IEEE International Symposium on Information Theory Proceedings (ISIT)*, 1064–1068.

Samet, H. 2005. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Sipser, M., and Spielman, D. A. 1996. Expander codes. *IEEE Trans. on Inf. Theory* 42(6):1710–1722.

Spielman, D. A.; Wang, H.; and Wright, J. 2012. Exact recovery of sparsely-used dictionaries. In *25th Annual Conference on Learning Theory (COLT)*. http://www.columbia.edu/ jw2966/papers/SWW12-pp.pdf.

Tanaka, F., and Edwards, S. F. 1980. Analytic theory of the ground state properties of a spin glass. i. ising spin glass. *Journal of Physics F: Metal Physics* 10(12):2769.

Wang, J.; Shen, H. T.; Song, J.; and Ji, J. 2014. Hashing for similarity search: A survey. *CoRR* abs/1408.2927.

Yu, C.; Gripon, V.; Jiang, X.; and Jégou, H. 2015. Neural associative memories as accelerators for binary vector search. In *Proceedings of Cognitive*, 85–89.