Clique Based Neural Associative Memories with Local Coding and Pre-Coding

Asieh Abolpour Mofrad

Asieh.Mofrad@uib.no

The Selmer Center, Dept. of Informatics, University of Bergen, Bergen, Norway

Matthew G. Parker

Matthew.Parker@ii.uib.no

The Selmer Center, Dept. of Informatics, University of Bergen, Bergen, Norway

Zahra Ferdosi

Ferdosi@aut.ac.ir

Dept. of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran

Mohammad H. Tadayon

Tadayon@itrc.ac.ir

Iran Telecommunication Research Center (ITRC), Tehran, Iran

Keywords: Neural associative memory, error correcting codes, local coding, pre-coding

Abstract

Techniques from coding theory are able to improve the efficiency of neuro-inspired and neural associative memories by forcing some construction and constraints on the network. In this article the approach is to embed coding techniques into neural associative memory in order to increase their performance in the presence of partial erasures. The motivation comes from the recent works by Gripon, Berrou and co-authors, which revisited Willshaw networks and presented a neural network with interacting neurons that partitioned into clusters. The model introduced stores patterns as cliques of small size which can be retrieved in spite of partial error. We focus on improving the success of retrieval by applying two techniques; first by doing a local coding in each cluster and then by applying a pre-coding step. We use a slightly different decoding scheme, which is appropriate for partial erasures and converges faster. Although the idea of local coding and pre-coding are not new, the way we apply them is different. Simulations show an increase in the pattern retrieval capacity for both techniques. Moreover we use self-dual additive codes over field $GF(4)$ which have very interesting properties and a simple-graph representation.

1 Introduction

Neural associative memory is capable of memorising (learning) a set of patterns and retrieving the full matching pattern from a given noisy fragment of it. This functionality is similar to communication over a noisy channel. Channel coding concerns reliable and efficient retrieval of a set of patterns (codewords in coding theory terminology) from a noisy version that the receiver receives. Generally speaking for digital error correcting codes, a subset of all possible pattern configurations is chosen for transmission. Coding techniques concern choosing this subset so that the receiver, which knows the allowed patterns (codewords), can figure out whether the received pattern is an allowed one and in the case of non-allowed patterns finds the closest allowed pattern (i.e. decodes the received word to the most likely codeword sent). Therefore codes are carefully constructed to have high efficiency in the sense that the noise distance between pairs of codewords is as large as possible given the size of the codeset. On the other hand neural associative memories are generally able to memorise any set of randomly chosen patterns and as a consequence they are not optimised for noise distance. Researchers who have applied coding theory to neural associative memories include (Hopfield, 2008; Berrou and Gripon, 2010; Gripon, 2011; Salavati, 2014; Berrou et al., 2014). One approach in this context is to focus on learning patterns that have some sort of inherent redundancy - in another approach the network is designed to be able to memorise any random set of patterns. For instance Berrou and Gripon (Berrou and Gripon, 2010) achieved considerable improvements in the pattern retrieval capacity of Hopfield networks, by utilising Walsh-Hadamard sequences. Salavati et al proposed a neural association mechanism that employs binary neurons to memorise patterns belonging to

another family of low correlation sequences, called Gold sequences (Salavati et al., 2011). In (Boguslawski et al., 2014) some strategies to store non-uniform patterns, such as by adding random bits and using Huffman coding- which is a data compression technique- are discussed.

Dividing a learnt pattern into sub-patterns can be shown to be useful in several ways, see for instance (Berrou and Gripon, 2010; Hopfield, 2008; Gripon and Berrou, 2011; Salavati and Karbasi, 2012), and for more details see (Gripon, 2011) and (Salavati, 2014). This approach also limits the allowed pattern configurations.

We follow the neural structure introduced in (Gripon and Berrou, 2011). These neural structures (called the GB model hereafter), which are based on Willshaw networks (Willshaw et al., 1969), are formed by dividing a neural network with n neurons into c clusters of size n/c each. The patterns are then chosen so that only one neuron in each cluster is active for a given pattern. Therefore a pattern can be considered as a random vector of length $c\log(n/c)$, where the $\log(n/c)$ part specifies the index of the active neuron in a given cluster. To memorise a pattern one then forms edges between active neurons and makes a clique (complete sub-graphs) of order c. The decoding process is then to retrieve the erased nodes of the clique using edges stored during learning.

It is worth mentioning that in (Hopfield, 2008) there is a model of an associative memory developed within a biological setting. In this model neurons (n) are partitioned into a number of categories (say c) with n/c possible values. A pattern then gets a single value in each category -the cluster counterpart to the GB model - and like the GB model learning a new pattern is achieved by establishing edges between active neurons. Al-

though the topology and learning part are similar, the retrieval part is different. There are other major differences that may be interesting to study because the Hopfield model focuses more on the biological aspects whereas the GB model arises from coding techniques. For instance, for about the same number of neurons the number of categories in the simulations is much larger than the number of clusters and consequently the number of neurons in each Hopfield category is much less than in GB clusters - as an example compare 50 categories, each with 20 neurons, vs. 4 clusters, each with 256 neurons.
In the Hopfield model the pattern set is generated by randomly choosing a neuron in each category, according to a power law distribution ($p(n) \sim \frac{1}{n^{1/2}}$), whilst in the GB model active neurons in clusters are independent and identically distributed (i.i.d.). As mentioned previously, non-uniform distributions are also considered for the GB model (Boguslawski et al., 2014). Moreover, the Hamming distance between two patterns in the Hopfield model is defined as the number of neurons in which they differ, whilst in the GB model the Hamming distance is the number of clique edges in which two patterns differ, which means that distance is far better for the latter case.

The GB based models proposed in this article focus on improving storage performance and making memory more resistant in the presence of partial erasure. Both local coding and pre-coding are techniques used to enhance pattern retrieval capacity and have been used in neural associative memory. For instance clustering the neurons and applying the rule that just one neuron in each cluster is allowed to be active is itself a local coding (Hopfield, 2008; Gripon and Berrou, 2011). Another example is a two-level neural associative model in (Salavati and Karbasi, 2012) in which the pattern neurons are divided into clusters and each cluster is a bipartite graph - inspired from

graph-based codes like LDPC (low density parity check) codes - where sub-patterns should form a subspace - a code in coding terminology. The second level may enforce constraints in the same sub-pattern space - just local coding - or in a totally different space - a combination of local coding and pre-coding.

The local coding construction proposed in this paper does not affect the number of neurons but adds redundancy to the patterns and then learns codewords assigned to each sub-pattern in the neural network. Part of this work was presented at CWIT (Mofrad et al., 2015) and here we improve on the decoding algorithm introduced there to make it suitable for partial erasures, and this reduces the size of the neurons involved in the retrieval process, and thus they converge faster especially in the context of iterative decoding.

The pre-coding technique is a more straightforward way to improve the pattern retrieval capacity and there is an argument that working with structured patterns is biologically meaningful and that sensory inputs to the brain are pre-processed before actually being stored, (Salavati et al., 2011; Salavati, 2014; Berrou et al., 2014).

The pre-coding technique that we consider simply encodes the patterns and then splits the corresponding codewords and memorises each part in a cluster. We perform experiments in the presence of partial erasure and compare local coding and pre-coding models - these two schemes can then be combined if one needs more data protection.

For simulation we select two error correcting codes; the algebraic Reed-Solomon (RS) code, which is a maximum distance separable (MDS) linear code (MDS means that, for a fixed codeword length n, and pattern length k, then MDS codes have the greatest error correcting and detecting capabilities). RS codes are widely used for data

storage and are suitable for erasure errors (Reed and Solomon, 1960).

The second class of error correcting code that we select is the self-dual additive codes over $GF(4)$ see (Danielsen, 2008) and references therein. These codes can be represented as simple graphs and have many interesting features. As far as we know the graph-based codes that have been used in neuro-inspired memories in the literature have bipartite representation - see (Salavati, 2014) and (Berrou and Gripon, 2010) for instance. Although in this article we do not consider the graphical representation of these codes and just consider them as a second error correcting code because these codes have more flexible parameters suited to the design of the network, in future work we shall apply message-passing algorithms to the simple graphs representing these $GF(4)$-additive codes to improve decoding performance.

The rest of the paper is as follows: Section 2 reviews the basics and the clique-based networks introduced by Gripon and Berrou -notations from (Gripon and Berrou, 2011) mostly. Section 3 is devoted to the local coding scheme and pre-coding model. In Section 4 our decoding algorithm is explained by an example. Section 5 contains the simulation results and a comparison of neural networks both with and without local coding. The results for local coding and pre-coding are also compared and discussed. Section 6 concludes, and the detailed decoding algorithm is provided in the appendix.

2 GB model of neural networks

Gripon and Berrou introduced a model where, by splitting a network of n neurons into c clusters of size $l = n/c$, any alphabet (say $\mathcal{A}$) with cardinality $|\mathcal{A}| = l$ can be

depicted. The model allows for different size alphabets and clusters but, for simplicity, is considered fixed with $l = 2^\kappa$, so as to ease working with binary patterns. Each binary pattern of length κ is then assigned to a unique neuron or equivalently to a character of alphabet $\mathcal{A}$:

$$f : \{0,1\}^\kappa \to [|1; l|].$$

where $[|1; l|]$ is the subset of integers between 1 and l.

The learning process is simply to store patterns of length $k = c\kappa$ as cliques of size c where a unique neuron is selected from each individual cluster. More formally consider learning pattern m:

$$C \ : \ m = m_1 m_2 \cdots m_c \to (f(m_1), f(m_2), \cdots, f(m_c))$$

where each $m_i \in \{0,1\}^\kappa$, $1 \leq i \leq c$ is a binary pattern of size κ. The active neurons, $f(m_i)$, connect together by edges to make a clique, as in Fig. 1. The value of each neuron is considered binary, i.e. if a node is within a clique for a given pattern, its value is 1, and 0 otherwise. If $W(m)$ denotes the set of edges of the clique for pattern m, then the edges after learning a set of patterns, M, will be:

$$W = \bigcup_{m \in M} W(m)$$

Retrieving or recalling part of a learned pattern is done in two steps and can be iterative. The algorithm finds the most probable active neuron in each cluster.

See (ABOUDIB et al., 2014; JIANG, 2014) for a detailed study of the retrieval algorithm. For instance, the different approaches of Global Winners-Take-All (GWsTA) and Global Losers-Kicked-Out (GLsKO) both improve the retrieval performance. Our decoding is designed for partial erasures and reduces computations.

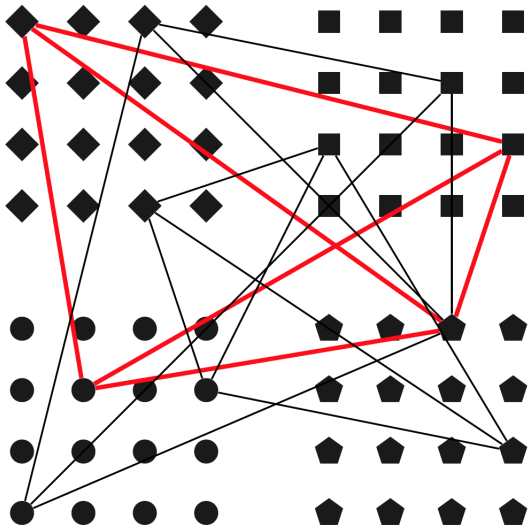


Figure 1: Learning process in a network with 64 neurons which split into four clusters of 16 neurons each. Red edges represent the binary pattern $0000, 1011, 0101, 0010$ which is learned as a clique.

3 Local coding technique and a pre-coding clique-based model

As mentioned in the introduction, the GB model inherently has a local coding in which the allowed sub-patterns are those with exactly a 1 in their binary representation (a kind of constant weight code in each cluster). However, our idea for local coding is to map a codeword instead of a sub-pattern to each neuron. English language is a good example to explain our local coding technique. Consider the set of learning patterns consisting of meaningful sentences with a fixed length (i.e. each with the same number of words, for instance consider as a sample this quote from Nelson Mandela as a pattern to be memorised; "A winner is a dreamer who never gives up"). So the network we choose has 9 clusters and there is a one-to-one map between all possible words (sub-patterns)

and the neurons in each cluster. A partial erasure then is like "A w-n–r i- - dr—— w-o n–er giv-s up" and the local retrieval deals with spelling of the words and meaningful words - well separated codewords in the model - and in the higher level the grammar or the meaning of the sentence is checked - clique connections in the model. The local coding technique in this example is implemented in terms of those words that are allowed in the sentence, i.e. codewords in the model are allowed words in this example.

Local codes can be chosen from different alphabets, rates and minimum Hamming distances -and it is possible to consider different codes with different codebook sizes for each cluster. The Hamming distance between two words of the same length - or codewords - is the number of positions with different symbols. The minimum distance of a code [1], is the lowest Hamming distance between any two codewords in the code. If we choose a code with high minimum distance and a partial erasure happens, then the minimum distance of a local code may eliminate that erasure and then the ordinary decoding of GB neural networks can be done more efficiently.

More formally, consider that the goal is to learn patterns of type $m = m_1 m_2 \cdots m_c$ where each m_i, $1 \leq i \leq c$ is a non-binary pattern of size κ. Components of m_i can be binary as well, but we choose them from the finite field $GF(2^p)$ and use an algebraic Reed-Solomon (RS) code, which is a maximum distance separable (MDS) linear code and has the best possible minimum distance. Therefore a neural network of c clusters, each with $l = 2^{p\kappa}$ neurons, can represent patterns like m. Recall that if no local coding is done then each sub-pattern m_i maps to neuron $f(m_i)$ in the ith cluster:

$$m = m_1 m_2 \cdots m_c \rightarrow (f(m_1), f(m_2), \cdots, f(m_c))$$

[1] Minimum distance is a very important parameter in designing block codes

where $f : GF(2^p)^\kappa \rightarrow [|1; l|]$.

Linear codes, like RS codes, have a generator matrix whose rows form a basis for them. So codewords of a code $\mathcal{C}$ with generator matrix G have codewords like $g(m_i) = m_i G$ for each sub-pattern m_i. Then m maps to $m_g = g(m_1)g(m_2)\cdots g(m_c)$ and $f : \mathcal{C} \rightarrow [|1; l|]$ maps a codeword to a neuron and in general:

$$m = m_1 m_2 \cdots m_c \rightarrow (f(g(m_1)), f(g(m_2)), \cdots, f(g(m_c)))$$

As a toy example let $l = 16$ and the local code be a Hamming code (Hamming, 1950) $(7, 4)$ i.e. a binary sub-pattern $m_i = (m_i^1 m_i^2 m_i^3 m_i^4)$ is coded into $g(m_i) = (p_1 p_2 m_i^1 p_3 m_i^2 m_i^3 m_i^4)$ where $p_1 = m_i^1 + m_i^2 + m_i^4$, $p_2 = m_i^1 + m_i^3 + m_i^4$ and $p_3 = m_i^2 + m_i^3 + m_i^4$ where all additions are modulo 2. Then G is:

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}. \tag{1}$$

and it can be seen easily that $g(m_i) = m_i G$. See Fig. 2 for the local coding scheme using Hamming code $(7, 4)$.

3.1 Pre-Coding clique-based neural networks

We recruit another example from language to explain the pre-coding technique to make comparison between local coding and pre-coding easier. Consider the patterns after pre-coding be a set of meaningful words -codewords in the model- with the same number of syllables -the number of clusters in the network. A syllable, which may or may not have

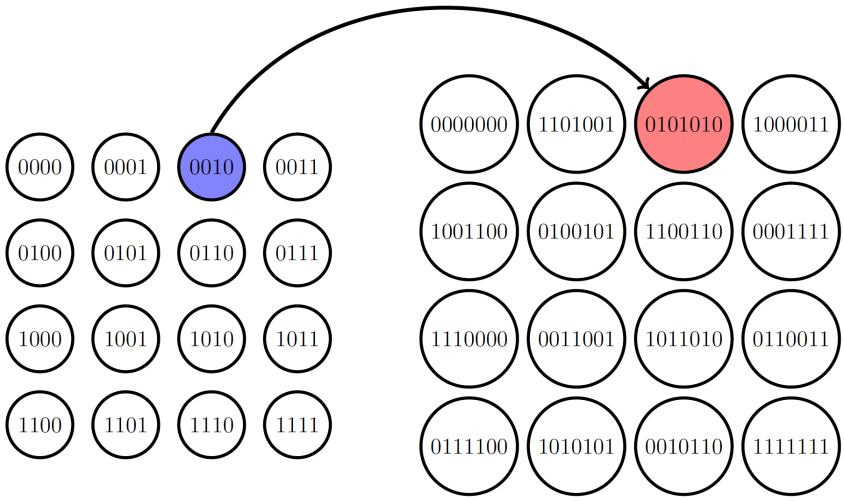


Figure 2: Each neuron in the cluster is assigned to a sub-pattern before local coding (left) and to a codeword of Hamming code $(7, 4)$ after local coding (right)

a meaning, is made up of phonemes and each neuron represents a syllable. Because a phoneme is the smallest unit of sound that distinguishes one word from another, we can consider them as the alphabet used in the pre-coding. For instance to memorize "astronomical", /æs.trə.nɑ.mɪ.kəl/, we need 5 clusters and in each cluster we have a one-to-one map between all syllables and neurons[2]. On recalling, a clue like /æ-.t-ə.--.m-.kəl/ may be given. Although there is not a meaning (a particular minimum distance), syllables are not a random combination of phonemes and some degree of regularity holds which facilitates erasure correcting in clusters. The edges established to make cliques in this example can represent the spelling or meaning for instance. In this example the role of cliques is more important and the distance between cliques is greater.

[2]The length of syllables is not important in this example, but as the length is fixed in the model, one can consider a fixed 3 phoneme for all neurons and add an empty sign to those syllables with less phonemes

We compare local coding with pre-coding in Section 5 by results from simulations.

4 Recalling from a partially erased pattern

We receive some partially erased pattern from which erased symbols must be retrieved. Equivalently in clique-based models we must find a clique that contains the provided symbols as active neurons -i.e. neurons whose value is 1. As the given part of a learnt pattern is assumed correct, then recalling is simply a matter of finding a match from memorized patterns. To avoid unnecessary computation, we introduce a decoding algorithm suitable for retrieving partial erasures. We explain the two level retrieval algorithm by examples from English language provided in Section 3 and a formal version of the algorithm can be found in the Appendix.

Suppose from the partially erased sentence "A w-n— i- - dr—— w-o n–er giv-s up", the memory tries to recall the complete sentence. The first step is the local search within the clusters for all possible words that match. If there is a unique option, like 'A' in the first cluster and 'gives up' in the last cluster -we suppose there are no other words that match 'giv-s up'- the corresponding neuron is active and all edges contained in the learned edge set W with one end point at these active neurons are established. Suppose for the second cluster there are candidate words (neurons): {'window', 'winner', 'winter', 'winrar', 'wonder'}. Similarly for the third cluster: {'id','if', 'in', 'is', 'it' }; fourth cluster: {'A','I'}; fifth cluster: {'dragoon', 'dreamed', 'dreamer', 'driving, 'drunken}; sixth cluster: {'who', 'woo'}; and the seventh cluster: {'nagger', 'nailer', 'never', 'number'}. A better minimum distance in local coding reduces the size of these

candidate sets. The second level then checks the degree of each word. Starting from the second cluster, suppose the degree of 'wonder' is zero, and the degree of 'window' and 'winrar' is 1 and the degree of 'winner' and 'winter' is 2. So from the sentence (cliques) information, we know that just two valid words remain at the second position (cluster). The word candidates are: {'winner','winter'}. By the same argument suppose new sets update as follows; third cluster: {'if', 'is' }; fourth cluster: {'A'}; fifth cluster: { 'dreamer', 'drunken'}; sixth cluster: {'who'}; and the seventh cluster: {'never'}. So the active neurons in the fourth, sixth and seventh cluster are found and the algorithm repeats by establishing edges from these three neurons and checking the degree of each to remove the ones whose degree is less than 5. This recall may be successfully finished after one or two more iterations. But consider the case where we end up with the sentence "A winner is a dr—— who never gives up" and both remaining candidates, i.e. 'dreamer' and 'drunken', have degree 7. In this case recalling fails. These kinds of failure would happen because there is no rule that forbids too similar sentences being members of the learning set. More formally, sub-patterns are chosen randomly and although the clique form plays the role of grammar or meaning for instance, sentences that are too close may still cause problems. In comparison, such a problem will not happen with the pre-coding technique because the learning set patterns have a high pairwise minimum distance.

Overall, a good local coding limits the possible matching set[3], on the other hand a pre-coding forces patterns to be well separated. The best strategy is to use both techniques

[3]For instance 'who' and 'woo' have Hamming distance 1 and in a good coding scheme both can not be codewords simultaneously

together to have a more reliable memory.

For the last example let the pattern set be all sentences of length c, with at least d_p different words between any pair of sentences, and at least d_l different letters between any two words in the same position. The condition is strict but the greater d_p and d_p can be made, the more reliable the retrieval.

5 Simulation Results

To see the performance of the proposed associative memory with local coding we first consider a network of 4096 neurons that are clustered in 8 sets, each with 512 neurons. For local coding the $[7, 3, 5]_8$ RS code is used, i.e. with this code any sub-pattern of length 3 where its components are taken from $GF(8)$, maps to a codeword of length 7 so that the minimum Hamming distance of the new set of codewords is 5. By fixing the learning set size, we see the results for different partial erasure probabilities in Fig. 3, and the retrieval performance when erasure probability is fixed and learning set size is growing is shown in Fig. 4.[4]

See Fig. 5 and Fig. 6 for the same comparison in a network of 512 neurons that are clustered in 8 sets each with 64 neurons when the local coding is the $(6, 2^6, 4)$ Hexacode, (see Conway and Sloane, 1988, for instance). The Hexacode is a self-dual $GF(4)$ additive code and so can be represented by a simple graph. Its generator matrix

[4] As can be seen, the performance obtained with the proposed local-coding is dramatically better than the uncoded performance. The main cost of this performance is an increased word length (3 symbols in $GF(8)$ to 7 symbols in the same field) for each node.

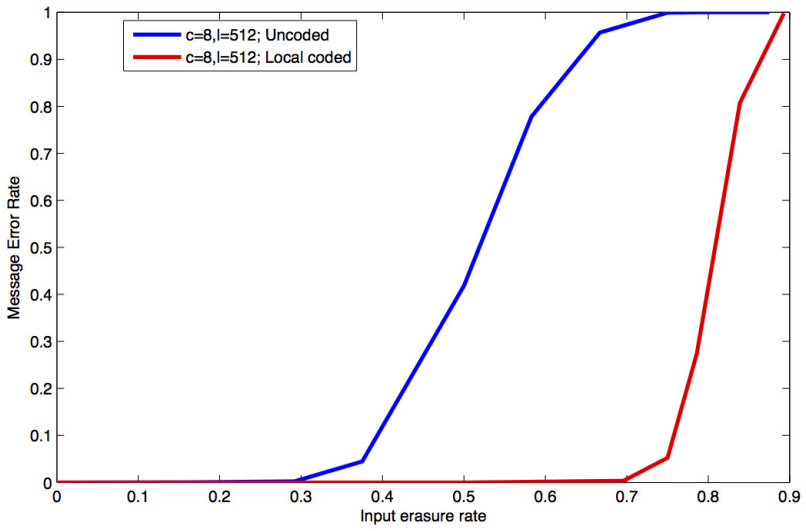
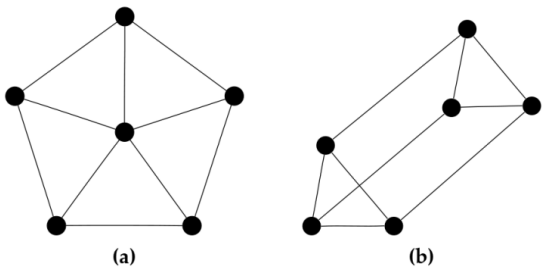


Figure 3: Comparison of performance of an uncoded associative memory of 8 clusters, each with 512 neurons (blue curve), with the coded version (red curve) where local coding uses the $[7, 3, 5]_8$ RS code for $|M| = 50000$.

corresponding to graph (b) is [5]:

$$G = \begin{pmatrix} \omega & 1 & 1 & 1 & 0 & 0 \\ 1 & \omega & 1 & 0 & 1 & 0 \\ 1 & 1 & \omega & 0 & 0 & 1 \\ 1 & 0 & 0 & \omega & 1 & 1 \\ 0 & 1 & 0 & 1 & \omega & 1 \\ 0 & 0 & 1 & 1 & 1 & \omega \end{pmatrix}$$



Two graph representations of the hexacode (Danielsen, 2008)

[5] The generator matrix is obtained from the adjacency matrix of the graph, (b), by setting all the diagonal entries to ω.

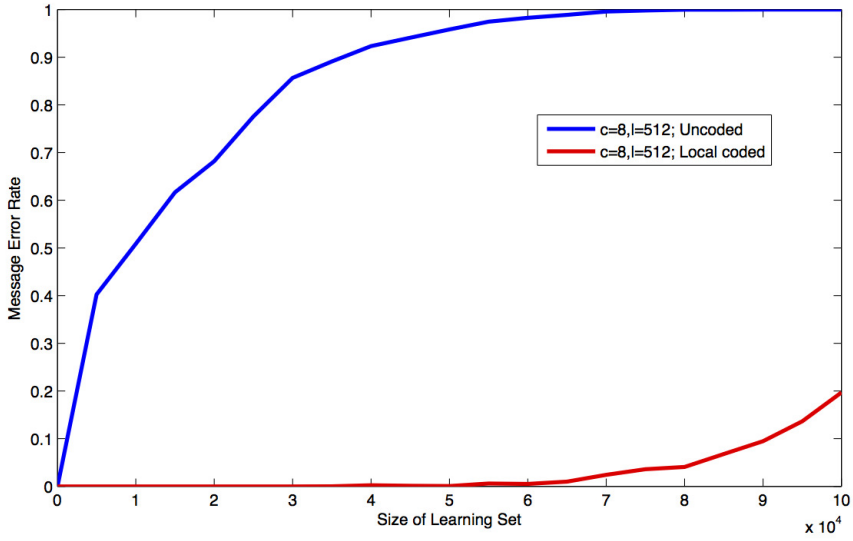


Figure 4: Comparison of performance for an uncoded associative memory of 8 clusters, each with 512 neurons (blue curve), with the coded version where local coding uses the $[7, 3, 5]_8$ RS code (red curve). The erasure rate for each symbol is 0.7. The largest dataset here is 100000.

As opposed to the RS code, the number of neurons using local coding with the Hexacode does not change, i.e. the length of each sub-pattern remains fixed but the field changes from $GF(2)$ to $GF(4)$.[6]

As mentioned in the Introduction, we propose to use self-dual $GF(4)$ additive codes as their parameters are more flexible. Moreover, we also intend to use such codes in our future works because of their graphical representations. In particular it is known that nested-clique graphs represent many of the strongest $GF(4)$ additive codes in terms of pairwise distance and optimum edge sparsity and are therefore good candidates from which to build nested-clique neural networks. The idea would be to embed a self-dual

[6]The hexacode is an additive code, i.e. a binary vector m_i (sub-pattern for local coding) generates all 2^6 codewords by m_iG (i.e. m_i is taken over $GF(2)$ not $GF(4)$).

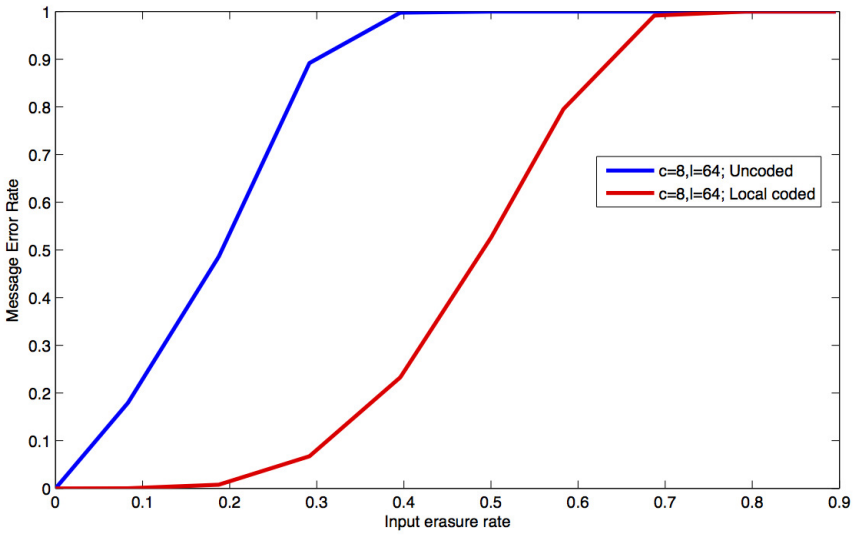


Figure 5: Comparison of performance of an uncoded associative memory of 8 clusters, each with 64 neurons (blue curve), with the coded version where local coding uses the hexacode $(6, 2^6, 4)$ for $|M| = 4000$.

code inside each neuron and benefit from this graph code during the retrieval process.

For the pre-coding technique we choose a $(12, 2^{12}, 6)$ self-dual $GF(4)$ additive code (the dodecacode) (Calderbank et al., 1998). This code maps any binary pattern of size 12 to a codeword of size 12 in $GF(4)$ with minimum distance 6. We have $c = 4$ clusters each with $l = 64$ neurons and the length of each sub-pattern is 3. This is compared to an uncoded version as well as to a local coding version. For the local coding we use the hexacode again but the number of clusters is set to be 4.

Fig 7 and 8 confirm the expectation that pre-coding improves the capability of memorising a larger set of patterns and to recall successfully in the presence of stronger partial erasure. From Fig. 7 we see that when the erasure rate is smaller than 0.4 the

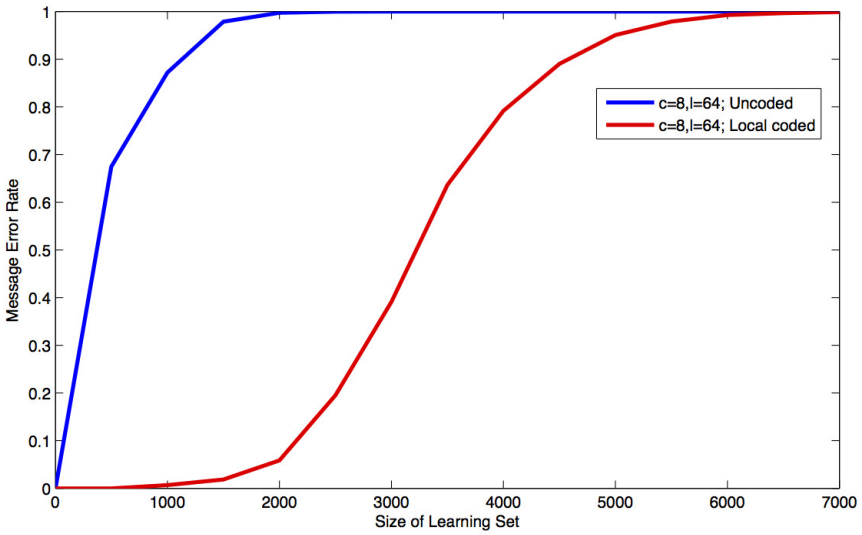


Figure 6: Comparison of performance for an uncoded associative memory of 8 clusters, each with 64 neurons (blue curve), with the coded version where local coding uses the hexacode $(6, 2^6, 4)$. The erasure rate for each symbol is 0.6.

pre-coding technique gives better results. But in the case of higher erasure probability, local coding outperforms. This is justified by the following argument: after the first check, whenever partial erasure is low, the number of clusters with an active neuron is large. So the pre-coding technique is able to benefit more from its minimum distance than that associated with the distance between cliques. On the other hand, when erasure probability is higher, the ability of local retrieval is more important, which is what the local coding model was designed for.

Then to compare storage capacity, we fix erasure at 0.4 -where both techniques showed similar error rate in retrieval. Fig. 8 shows that when the learning set is smaller then local coding technique performs better, whilst for larger learning sets the pre-coding technique outperforms. Again, this result is as expected. For a fairly small learning set

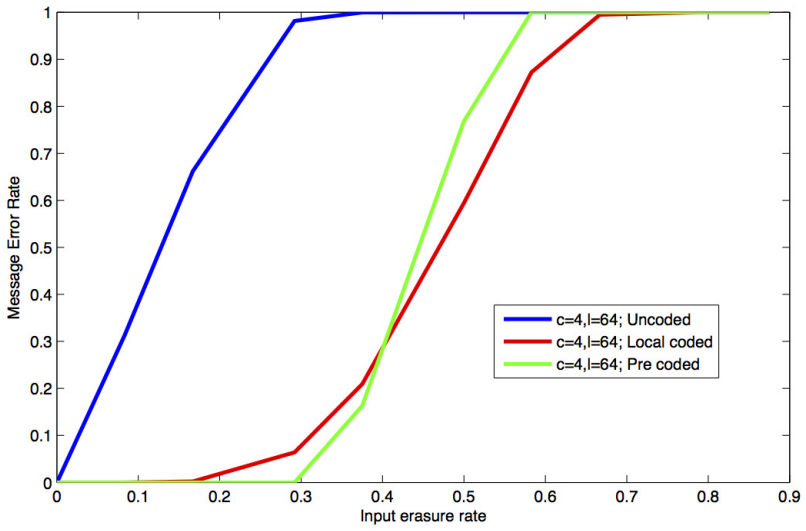


Figure 7: Comparison of performance of an uncoded associative memory of 4 clusters, each with 64 neurons (blue curve), with the local coded version where local coding uses the hexacode $(6, 2^6, 4)$ (red curve) and is pre-coded with the $(12, 2^{12}, 6)$ code (green curve) for $|M| = 3000$.

the number of edges is smaller and the cliques are more likely to have a higher minimum distance, so the role of local coding is more important. In contrast, when the number of edges is increased by increasing the size of the learning set, the role of pre-coding and minimum distance amongst the cliques become more important.

Note that in all simulations the learning part is done independently of whether local coding is done or not. Indeed the symbols in the patterns are considered i.i.d. random variables. If we do local coding then the set of edges is exactly the same, but pre-coding changes the shape of cliques, so W is not the same any more.

As the data set is chosen randomly, we repeat the experiment 100 times and com-

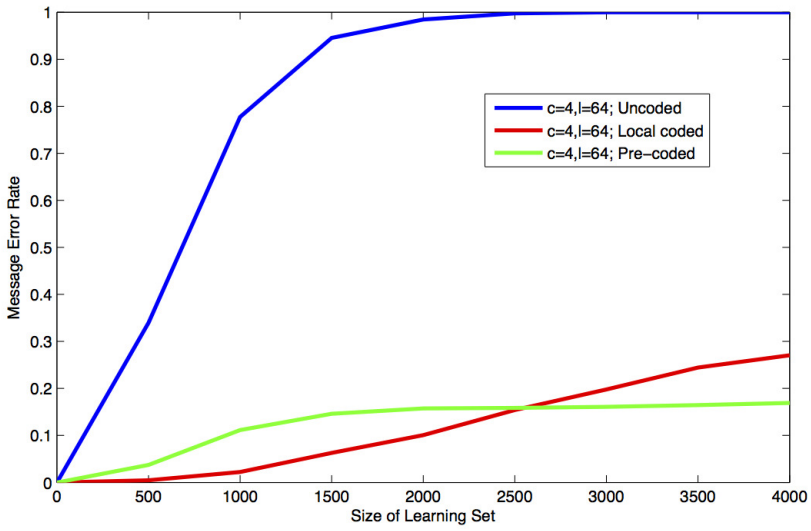


Figure 8: A comparison of the performance for an uncoded associative memory of 4 clusters, each with 64 neurons (blue curve), with the local coded version where local coding uses the hexacode $(6, 2^6, 4)$ (red curve) and pre-coded with the $(12, 2^{12}, 6)$ code (green curve). The erasure rate for each symbol is 0.4.

pute the average to have more reliable results (i.e. choose 100 random patterns from the learning set as the input and partially erase it). Again, for any erasure probability, the symbols to be erased are chosen randomly, so we partially erased each pattern by 100 different randomly chosen erasure vectors. We then tried retrieving the chosen pattern and if the pattern is completely retrieved the algorithm is successful, otherwise it fails. Finally, the ratio can be computed for each data set and an average taken over 100 different erasure vectors.

6 Conclusion

Some applications of coding techniques in neuro-inspired associative memories have been discussed, and we have shown that both the local coding and pre-coding models based on the GB model have excellent error performance in the presence of partial erasures. The results are somewhat theoretical but, due to their structure and ability to retrieve patterns from a partial clue, such memories have potential application to content-addressable memories and to search engine algorithms. Our simulation results suggest that the local coding model is more suited to the case where the erasure probability is high and/or the learning set is pretty small. In contrast, the pre-coding model seems to be more suited to the situation where the erasure probability is not that high and/or the size of the learning set is rather large. A new version of the decoding algorithm is presented which reduces the computational complexity and is suitable for partial erasures.

It is not necessary that the local coding be non-binary or use extension fields over $GF(2^p)$. We chose RS codes because they are suitable for storage and erasure type errors. We also considered self-dual $GF(4)$ additive codes as we shall exploit their graph representations in future work.

Appendix

The detailed version of the recalling algorithm is provided here. It assumes the neural network has local coding but it can also be used for an uncoded version, similar to a

pre-coding model as well. We assume that the same code $\mathcal{C}$ is used in all clusters.

Consider that a noisy version of a learned pattern

$$m_g = (m_{1_1} m_{1_2} \cdots m_{1_t} | m_{2_1} m_{2_2} \cdots m_{2_t} | \cdots | m_{c_1} m_{c_2} \cdots m_{c_t})$$

is $\hat{m}_g = (\hat{m}_{1_1} \hat{m}_{1_2} \cdots \hat{m}_{1_t} | \hat{m}_{2_1} \hat{m}_{2_2} \cdots \hat{m}_{2_t} | \cdots | \hat{m}_{c_1} \hat{m}_{c_2} \cdots \hat{m}_{c_t})$ where symbol m_{i_r} is given: $\hat{m}_{i_r} = m_{i_r}$, or is erased: $\hat{m}_{i_r} = e$, and t is the length of the codeword, so $t > \kappa$ if we have local coding and $t = \kappa$ otherwise.

We also assign $\hat{m}_i = m_i$ iff all the m_{i_r} are known and $\hat{m}_i = e$ iff at least, for one r, $\hat{m}_{i_r} = e$.

To begin, we separate clusters into two sets C_u and C_e for unerrored and errored components, respectively:

$$C_u = \{i : \hat{m}_i = m_i\} \text{ and } C_e = \{i : \hat{m}_i = e\}, 1 \leq i \leq c.$$

Each neuron is shown with $n_{ij}, 1 \leq i \leq n, 1 \leq j \leq l$, that is equivalent to a unique value in $\{0, 1, 2, \cdots, l-1\}$. So n_{ij} is a node in the graphical representation of m_i and for a specific j in cluster i we assign $n_{ij} = f(m_i)$. Note that m_i is a codeword here.

Also for $i \in C_e$ we define and initialize sets $T(i) = \{n_{ij} | \hat{m}_{i_r} = m_{i_r} \text{ or } \hat{m}_{i_r} = e;\ \forall j, r\}$. As we will see these sets play an important role in reducing computational complexity.

Once we construct all $T(i)$ sets a local check is done as follows:

For all $i \in C_e$; if $T(i) = \{n_{ij}\}$ for some j then

- Let $C_u = C_u \cup \{i\}$ and $C_e = C_e \setminus \{i\}$,
- Correct $\hat{m}_i$ by putting $\hat{m}_i = f^{-1}(n_{ij})$

Indeed as the n_{ij} correspond to codewords of a code, some erasures were corrected by this local check.

Values of the neurons are defined as:

$$v(n_{ij}) = \begin{cases} 1 & \text{if } i \in C_u \text{ and } n_{ij} = f(m_i), \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

We establish all edges contained in the learned edge set W, so that at least one node for each edge has value 1. More formally we initialise the edge set $w = \{(n_{ij}, n_{i'j'}) \in W | v(n_{ij}) = 1 \text{ or } v(n_{i'j'}) = 1\}$ where $1 \leq i, i' \leq c, 1 \leq j, j' \leq l$.

Now we can start iterative retrieval, see Alg. 1. If the first part of the algorithm (until line 32) retrieves the original pattern then it stops, but if there are several candidate neurons in each cluster C_e then we search in W for edges with end nodes in the $T(i)$ that make a clique of size $|C_e|$. This happens when the partial erasure is high and distributed so the active neuron in most clusters is unknown. Note that the definition of w is changed for the second part of the recalling (line 34).

Algorithm 1 Retrieval algorithm

Require: Initialize C_e, C_u, $T(i)$ for $i \in C_e$, $d(n_{ij}) = deg(n_{ij})$ for all neurons in $T(i)$, $v(n_{ij})$, w, $v_{\max} = |C_u|$, $Flag = True$, $Counter = 0$, $Retrieval = Failed$

1: **while** $Flag = True$ **do** ▷ In this loop the Alg. removes elements from the $T(i)$ sets and finds an active neuron in cluster i whenever $|T(i)| = 1$

2: **if** $v_{\max} = c$ **then** ▷ $v_{\max} = c$ means all clusters have their unique active neuron, Alg. stops by setting $Flag = False$

3: $Retrieval = Succeed$

4: $Flag = False$

5: **return** $Retrieval$

6: **else if** $v_{\max} = 0$ **then** ▷ $v_{\max} = 0$ means no active neuron is found at the first stage, go to line 33

7: $Flag = False$

8: **else**

9: **for** $i \in C_e$ **do** ▷ The edges cause some candidates to be removed from $T(i)$

10: **for** $n_{ij} \in T(i)$ **do**

11: **if** $d(n_{ij}) \neq v_{\max}$ **then**

12: $T(i) \leftarrow T(i) \setminus \{n_{ij}\}$ ▷ i.e. a new active neuron is found

13: **end if**

14: **end for**

15: **if** $|T(i)| = 1$ (i.e. $T(i) = \{n_{ij}\}$ for one j) **then**

16: $v(n_{ij}) = 1$

17: $\hat{m}_i = f^{-1}(n_{ij})$

18: $w = w \cup \{(n_{ij}, n_{i'j'}) \in W\}$ ▷ New edges establish from new activated neurons, C_u and C_v will update

19: $C_u = C_u \cup \{i\}$

20: $C_e = C_e \setminus \{i\}$

21: $Counter = Counter + 1$

22: **end if**

23: **end for**

24: **end if**

25: **if** $Counter \geq 1$ **then** ▷ i.e. a new active neuron is found in this iteration

26: Update $d(n_{ij})$ based on updated w

27: $v_{\max} = |C_u|$

28: $Counter = 0$

29: **else**

30: $Flag = False$ ▷ i.e. exit while loop and go to line 33

31: **end if**

32: **end while**

33: **if** $v_{\max} < c$ **then** ▷ Now all the remaining $T(i)$ sets have more than one neuron, so the alg. searches for a clique among neurons in $T(i)$ sets by establishing edges with at least one end within candidates

34: $w = \{(n_{ij}, n_{i'j'}) \in W | n_{ij} \in T(i) \text{ or } n_{i'j'} \in T(i')\}$

35: where $i, i' \in C_e, 1 \leq j, j' \leq l$

36: $N_{i' \to i} = \{n_{ij} | (n_{ij}, n_{i'j'}) \in w, 1 \leq j, j' \leq l\}$

37: where $i \neq i'$ and $i, i' \in C_e$

$d(n_{ij}) = \sum_{i' \in C_e \setminus \{i\}} \chi_{N_{i' \to i}}(n_{ij})$ ▷ The characteristic function $\chi_{N_{i' \to i}}$ is used to show the number of connected clusters to n_{ij} by at least 1 candidate neuron. The number of connections from a specific cluster does not matter.

$Flag = True$

while Flag=True **do**

  if $v_{\max} = c$ **then**

    $Retrieval = Succeed$

    $Flag = False$

  else

    for $i \in C_e$ **do**

      for $n_{ij} \in T(i)$ **do**

        if $d(n_{ij}) \neq |C_e| - 1$ **then** ▷ Those neurones which are not connected to all clusters in C_e are removed from $T(i)$

          $T(i) \leftarrow T(i) \setminus \{n_{ij}\}$

        end if

      end for

      if $|T(i)| = 1$ (i.e. $T(i) = \{n_{ij}\}$) **then**

        ▷ An active neuron n_{ij} is found

        $v(n_{ij}) = 1$

        $\hat{m}_i = f^{-1}(n_{ij})$

        $w = w \setminus \{(n_{ij''}, n_{i'j'}) \in w | 1 \leq i' \leq c \text{ and } 1 \leq j', j'' \leq l\}$

        ▷ Updating w by removing edges from the newly activated neuron n_{ij}

$C_u = C_u \cup \{i\}$

$C_e = C_e \setminus \{i\}$

Update $N_{i \to i'}$ by new C_e and w sets

$Counter = Counter + 1$

end if

end for

if $Counter \geq 1$ **then** ▷ At least one neuron is activated

Update $d(n_{ij})$ based on updated $N_{i \to i'}$

$v_{\max} = |C_u|$

$Counter = 0$

else

$Flag = False$ ▷ No new neuron is activated in the last iteration

end if

end if

end while

end if

return $Retrieval$

References

ABOUDIB, A., GRIPON, V., and JIANG, X. (2014). A study of retrieval algorithms of sparse messages in networks of neural cliques. In *COGNITIVE 2014: the 6th International Conference on Advanced Cognitive Technologies and Applications*, pages 140–146.

Berrou, C., Dufor, O., Gripon, V., and Jiang, X. (2014). Information, noise, coding, modulation: What about the brain? In *Turbo Codes and Iterative Information Processing (ISTC), 2014 8th International Symposium on*, pages 167–172. IEEE.

Berrou, C. and Gripon, V. (2010). Coded hopfield networks. In *Turbo Codes and Iterative Information Processing (ISTC), 2010 6th International Symposium on*, pages 1–5. IEEE.

Boguslawski, B., Gripon, V., Seguin, F., and Heitzmann, F. (2014). Huffman coding for storing non-uniformly distributed messages in networks of neural cliques. In *AAAI 2014: the 28th Conference on Artificial Intelligence*, volume 1, pages 262–268.

Calderbank, A., Rains, E., Shor, P., and Sloane, N. (1998). Quantum error correction via codes over gf (4). *IEEE Transactions on Information Theory*, 44(4):1369–1387.

Conway, J. H. and Sloane, N. J. (1988). Sphere packings, lattices and groups, volume 290 of grundlehren der mathematischen wissenschaften.

Danielsen, L. E. (2008). *On Connections Between Graphs, Codes, Quantum States, and Boolean Functions*. PhD thesis, The University of Bergen.

Gripon, V. (2011). *Networks of neural cliques*. PhD thesis, Télécom Bretagne.

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

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160.

Hopfield, J. J. (2008). Searching for memories, sudoku, implicit check bits, and the iterative use of not-always-correct rapid neural computation. *Neural Computation*, 20(5):1119–1164.

JIANG, X. (2014). *Storing sequences in binary neural networks with high efficiency*. PhD thesis, Télécom Bretagne, Université de Bretagne Occidentale.

Mofrad, A. A., Ferdosi, Z., Parker, M. G., and Tadayon, M. H. (2015). Neural network associative memories with local coding. In *Information Theory (CWIT), 2015 IEEE 14th Canadian Workshop on*, pages 178–181. IEEE.

Reed, I. S. and Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304.

Salavati, A. H. (2014). *Coding theory and neural associative memories with exponential pattern retrieval capacity*. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE.

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

Salavati, A. H., Kumar, K. R., Shokrollahi, A., and Gerstner, W. (2011). Neural precoding increases the pattern retrieval capacity of hopfield and bidirectional associative memories. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 850–854. IEEE.

Willshaw, D. J., Buneman, O. P., and Longuet-Higgins, H. C. (1969). Non-holographic associative memory. *Nature*.