Progress Report
August 22 to September 1, 2010

Amir Hesam Salavati
E-mail: hesam.salavati@epfl.ch

Supervisor: Prof. Amin Shokrollahi
E-mail: amin.shokrollahi@epfl.ch

Algorithmics Laboratory (ALGO)
Ecole Polytechnique Federale de Lausanne (EPFL)

September 1, 2010

1 Introduction

In the past week, I was reading two recent papers on coding theory and neural networks [1], [2]. These are the papers by Berrou, one of the inventors of turbo codes, and his team. Neither of them are published yet. The first one is going to presented in ISTC 2010 and the second one is submitted to IEEE transaction on neural networks.

In a nutshell, the two papers introduce a new type of artificial neural network capable of doing pattern matching and error correction in presence of noise. Compared to original Hopfield networks, which are able to memorize a number of given patterns, the proposed networks here are more powerful. They can learn more patterns and recall them in presence of erasure, a property that Hopfield networks lack. The key item in their power is the sparseness of the suggested structure. In what follows, I explain the suggested approach in more details.

2 Coded Hopfield Networks [1]

This paper extends the concept of Hopfield networks to error correcting neural networks, similar to the work of Cruz [3]. Like [3], the structure of the encoder considered here is a complete weighted bipartite graph, as shown in figure 1. We have k message and L code bits.

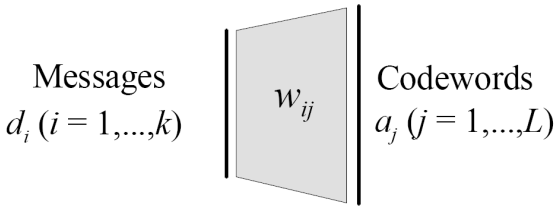

Figure 1: The neural bipartite graph of [1]

The weight between the i^{th} left node, d_i, and the j^{th} right node, a_j, is shown by w_{ij}. Weights are still determined as in original Hopfield networks,

according to equation (1).

$$w_{ij} = \sum_{m=1}^{M} d_i^m a_j^m \tag{1}$$

where M is the number of memorized patterns and d_i^m is the i^{th} bit of the m^{th} message. Likewise, a_j^m represents j^{th} bit of the m^{th} codeword. The set of a message-codeword correspondence makes one memorized pattern.

In contrast to [3], the proposed method does not achieve error correction by modifying learning weights. Instead, error correction is achieved by implementing a neural soft ML-decoder.

2.1 Recalling Process in Original Hopfield Networks

To recall a memorized pattern, Hopfield networks use the graph shown in figure 1 iteratively to reach a stable state.[1] In the beginning, left (right) nodes are initialized with received bits. Then, in each iteration, neurons update their state according to the weighted summed of their input. After a certain number of steps, the network becomes stable with the corresponding pattern given in the right (left) side of the graph.

In their original format, Hopfield networks can not perform error correction and retrieve the correct pattern in presence of noise. Therefore, authors propose a method to add an extra neural layer which performs ML-decoding to Hopfield networks to achieve error correction capabilities. The details of this scheme is given in the next section.

2.2 Soft Neural ML-decoder

Suppose we are given a codeword with some of its entries being erased. We would like to find out the correct pattern having this partial information. Having determined the correct codeword, we can then feed it on the righthand side of the network bipartite graph and retrieve the corresponding message.

In this paper, the authors propose a way to implement a neural soft ML-decoder. In the suggested method, a weighted bipartite graph with J left and Q right nodes are used. The left nodes correspond to the message bits.

[1] We have abused the notations slightly here. Originally, Hopfield networks are composed of a single complete graph with n nodes. However, such a structure could be mapped to a bipartite graph, similar to the one shown in figure 1.

Right nodes, called *fanals*, correspond to any one of the valid patterns, i.e. any one of the Q valid patterns is matched to one of the fanals. The graph is weighted and weights are also binary (± 1).

Weights are determined according to the mapping between fanals and messages. Let t_{jq} denote the weight between y_j (message nodes) and u_q (fanal). If, for instance u_q, represents the message "$1, 1, 1, -1, -1$", then $t_{1q} = t_{2q} = t_{3q} = 1$ and $t_{4q} = t_{5q} = -1$.

Equations (2) and (3) indicate the firing rules for message neurons and fanals, respectively. If none of the message bits are erased, then these equations ensure that only the correct fanal is active. Otherwise, more than one fanal is active, i.e. the message is erased, and the algorithm proceeds iteratively to correct the erasures.

$$y_j^p = \begin{cases} 1, & \text{If } v_j^p > 0; \\ -1, & \text{If } v_j^p < 0; \\ 0, & \text{If } v_j^p = 0; \end{cases} \tag{2}$$

In which $v_j^p = \sum_{q=1}^{Q} t_{jq} u_q^p$.

$$u_q^p = \begin{cases} 1, & \text{If } z_q^p = z_{max}^p \text{ and } z_{max}^p > \sigma; \\ 0, & \text{Otheriwse} \end{cases} \tag{3}$$

Where $z_q^p = \sum_{j=1}^{J} t_{jq}(y_j^p + \gamma y_j^{p-1})$ and $z_{max}^p = \max_q(z_p^q)$. In the above equations, p is the iteration number, σ is a fixed threshold and γ is a constant which models the memory effect. At $p = 1$, $y_j^1 = x_j,\ j = \{1, \ldots, J\}$, where x_j is the j^{th} bit of the received message. Note that to perform true ML decoding, σ must be equal to $-\infty$.

In words, here is what the decoder does:

1. On the right side, fanals calculate a weighted sum of the input plus the memory. Then the maximum of these sums are found. This value is compared to a threshold, σ. The corresponding fanal(s) with the maximum weighted sum fires a spike (send +1) to the message nodes if the sum is greater than the threshold. Otherwise, 0 is sent (and not -1).

2. On the left side, message neurons compute the weighted sum. They compare the sum with 0 and send the appropriate value based on the result (0,+1 or -1).

The second rule is similar to the Gallager's decoding algorithm for BEC: send erasure at variable's side if all of the check nodes are saying erasure. However, here the weighted sum is considered. So it's kind of weighted majority voting.

2.3 Results

The number of the messages which could be memorized and correctly recovered is a crucial parameter for the Hopfield network. The authors have compared this value for the original and modified Hopfield networks and show that there would a 22 to 75-fold improvement, depending on code length. Of course this comes at the price of complex decoder.

3 Sparse Neural Networks with Large Learning Diversity [2]

This paper introduces a more general framework compared to the first one. In essence, the goal is the same as before: we would like to design a neural network with error correction capability. In contrast to the previous work where a bipartite neural network was considered, here a network with n neurons are considered. This is more like the original Hopfield networks.

Here is how it works: we have n neurons which are divided into c clusters of size $l = n/c$. Let l be a power of 2 so that $\kappa = log_2(l)$ is the number of bits necessary to represent l. Neurons in each cluster are called *fanals* as only one of them could be active at any given time because of the learning process.

In contrast to original Hopfield network, the length of learned patterns is not n but $k < n$. We have a binary (± 1) message of length $k = c\kappa$ which uniquely identifies a pattern of fanals. Such active fanals are then fully connected in the graph. Hence, in the final graph, we have c clusters in which none of the nodes are connected to each other. On the other hands, there are only connections between the fanals of different clusters according to the learned messages. Each message corresponds to a unique fully connected subgraph. An example is shown in the figure 2. Note that this structure could be formally mapped to a weighted graph according to the following rule: if there is an edge between two nodes, the corresponding weight is equal to 1 and is 0 otherwise.

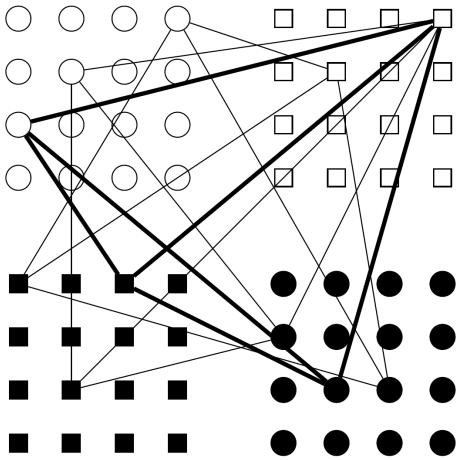

Figure 2: The neural multipartite graph of [2] with $c = 4$ and $l = 16$

3.1 Decoding Procedure

Now here comes the decoding problem: we have a message m of length $k = c\kappa$. We would like to determine the corresponding fanal pattern according to this message in presence of erasure, i.e. when several of k bits are erased. Therefore, from a given message, the network should retrieve the closest learned pattern.

The decoding procedure is performed in two steps, local and global:

- In the local step, given the input, the most likely fanal in a cluster is chosen.
- In the global step, having a set of active fanals, we decide which pattern is the closest fit. This pattern identifies the closest learned message.

3.1.1 Local Decoding

The local decoding process is the same method as discussed in the previous paper [1]. Equations (2) and (3) completely identify this procedure. In short, it is a *soft* ML-decoder which determines the most likely active fanal based on a given message or some part of it. If the provided information is sufficient,

the method comes up with only one fanal. Otherwise, it yields a number of possible fanals which means erasure, i.e. in this case the corresponding cluster could not tell which one its fanal must be active based on the received data.

3.1.2 Global Decoding

In this step, the algorithm retrieves the closest pattern given the decision of local decoders. Hence, the decoder first performs the local step for all clusters and then use global step to find the best pattern.

Let n_{ij} indicate the j^{th} fanal of the i^{th} cluster and $w_{(i'j')(ij)}$ be the weight between n_{ij} and $n_{i'j'}$. Denoting the output of neuron N by $v(N)$, the global decoding rule is given in equation (4).

$$v(n_{ij}) = \sum_{i'=1}^{c} \sum_{j'=1}^{l} w_{(i'j')(ij)} v(n_{i'j'}) + \gamma v(n_{ij}),\ \forall\ i, j \tag{4}$$

Where γ models the memory effect as before.

When considering inputs with a lot of noise or erasures, this algorithm can become iterative. However, a single iteration is enough is many cases.

4 Results

Authors have used analysis and simulation to assess the performance of their algorithm. The analysis is very limited in scope and provides only a worst case scenario, i.e. when only a single iteration of the algorithm is performed. In this case, and assuming after the local step c_e of c clusters are erased, i.e. could not decide which fanal must be active, then the approximated probability of error is:

$$P_e \approx l c_e (\frac{M}{l^2})^{c-c_e} \tag{5}$$

Figure 3 provides shows the simulation results when 4 out of 8 clusters were erased ($c_e = 4$, $c = 8$) and the decoding process was performed for 4 iterations (here $l = 256$). The performance obviously improves a lot compared to the case of only one iteration.

However, defining the code rate naturally as $R = \log_2(M)/n$, we notice that the code rate is quite poor. For instance, if an error rate of $P_e = 0.2$ is

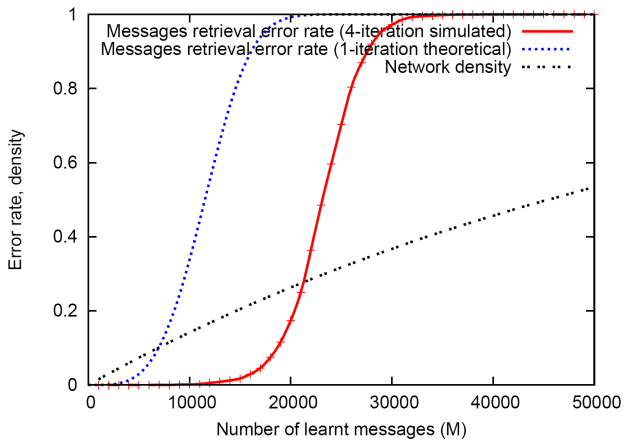

Figure 3: The performance of the algorithm for $c = 8$, $c_e = 4$ and $l = 256$. Graphs show the performance after one and four iterations [2].

acceptable, then we have $R = \log_2(20000)/(8 \times 256) = 0.007$. Albeit, note that here, half of the messages are erased (4 out of 8 clusters are erased). Hence, one might achieve better rates with less amount of erasure. In any case, the goal of the suggest method is not to provide us with high rate codes. Instead, it means to be a robust way of memorizing and recalling patterns in presence of noise. Table 4 confirms that the proposed algorithm is much superior to the standard Hopfield networks in this regards.

5 Some Issues And Future Works

This work is really interesting from a biological point of view as it addresses "sparse neural networks" in which only a few neurons are active at each time. Here, the neural network is divided to a few clusters and at any given moment, only one neuron in each cluster is active. This also shows some sort of "small words" effect. One problem with the proposed decoder in [1] and the local decoder in [2] though is that it is not local: at the fanals, you need the information for all check nodes to decide which one(s) should send 1 while others remain silent. This max function also makes the proposed algorithm

Model	HNN	Sparse network	ratio
Memory used	1.8×10^6	1.8×10^6	1
n	763	2048	$\times 2.7$
Message length	763	64	$\div 12$
Iterations	11	4	$\div 2.7$
Error probability	96%	2%	$\div 48$
Diversity	115	15000	$\times 130$
Capacity	8.8×10^4	9.6×10^5	$\times 12$

Figure 4: Performance of Hopfield networks compared to proposed algorithm in various aspects [2]

less biologically meaningful as locality is an important factor in real neural networks. In real neural networks, there are mechanisms implementing such max function but they are not fully identified yet.

Nevertheless, the suggested approach is a powerful method of storing and recalling patterns in presence of noise. Analyzing this method using the density evolution framework is one of the future works.

Another interesting topic would be to consider the weighted structure of the decoder mentioned in [2]. In the global step, weights are only binary (0/1). Extending the work to weighted graphs and analyzing the performance of such networks could result in more robust ways of storing patterns.

Overall, although [1] and [2] do not exactly address what we are interested to do, these works provide an interesting connection between coding theory and neural networks. Applying coding theoretical arguments in analyzing these artificial neural networks could help us extending the same approach to real neural systems.

References

[1] C. Berrou, V. Gripon, "Coded Hopfield Networks", Proc. Symposium on Turbo Codes and Iterative Information Processing, pp. 15, 2010.

[2] V. Gripon, C. Berrou, "Sparse Neural Networks with Large Learning Diversity", Submitted to IEEE Transaction on Neural Networks.

[3] D. She, J. B. Cruz, "Encoding Strategy for Maximum Noise Tolerance Bidirectional Associative Memory", IEEE Transactions On Neural Networks, Vol. 16, No. 2, 2005.