Progress Report
16-30 April 2012

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)

May 1, 2012

1 Summary

In the last two weeks, we presented our new model on neural associative memories to the group of Prof. Gerstner and they really liked it, specially its modularity, i.e. local connections among neurons. Having revived the green light from their group on the soundness of the model, Mr. Amin Karbasi and I have started analyzing the dynamic behavior of the network. Interestingly, it is very similar to the analysis of the *Peeling decoder* introduced by Luby et. al [1]. Therefore, I read that paper again more carefully to understand the way they analyze the decoding procedure. Although our neural algorithm is very similar to the operations done by the peeling decoder, they are exactly the same as each other. The preliminary analysis for our neural algorithm as well as the summary of [1] is explained in what follows.

Finally,I finished writing the first draft of the journal paper. Although simulation results are not complete yet, the text is finished.

2 Analysis of the proposed neural error correction method

For the sake of completeness, before we start explaining the analysis, the summary of the model is iven in the net section.

2.1 Network model and error correction

In the new model and problem formulation, we assume that we have a budget in terms of the number of constraint nodes we can have access to. Now the question is what is the best architecture which yields the lowest recall error probability with the given number of constraint nodes. To be more specific, assume we have N pattern nodes and a total of M constraint nodes. How should we spread these M constraint nodes to get the best performance? In [4], we used all the M constraints in a single layer, resulting in a bipartite neural graph capable of correcting at least one error in the input pattern. In [2], we spread the constraint nodes into two levels and further clustered them in the first level so that we ended up with local non-overlapping sub-network, each capable of correcting one bit of error. Therefore, we reduced the probability of error as the first level could correct multiple errors *if* they were distributed among separate local networks.

Following the same line of thought, one might think about different ways of clustering neurons so that errors get separated from each other. Here, we remove the second level and instead, distribute its constraints among the neurons of the first level. Therefore, we will have *overlapping local* connections. The model is illustrated in figure 1

In the learning phase, we follow the same learning approach mentioned in [2] or [3]. Therefore, we expect each sub-network be a sparse weighted graph. During the recall phase, we utilize the bit-flipping or the winner-take-all algorithms mentioned in [2]. In order to analyze the error correction performance of the proposed method during the recall phase, we follow a similar approach to *peeling decoders* in coding theory [1]:

1. If there is a cluster with only one bit of error in it, we assume it gets corrected after a few iterations. [1]

[1] We can follow a sequential approach in which the first sub-network performs one iteration of error correction, then the second one and so on. The advantage of this scheduling is that as long as error values are ± 1, single errors get corrected in one iteration

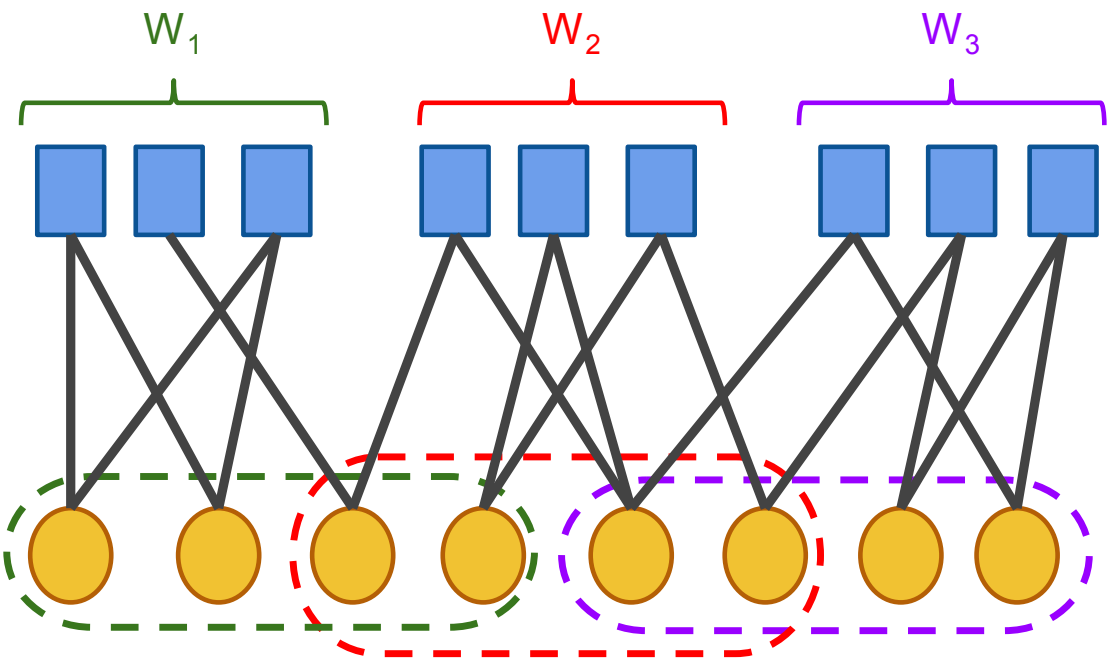

Figure 1: Overlapping clustered neural associative memory.

2. Removing the errors corrected in this round, we continue the above iteration until either all errors are corrected or no cluster with a single error remains, i.e. there are more than one error in all clusters. In the first case, we declare success, otherwise we have failed to correct the errors.

The advantage of the above model is its simplicity and a very rich analytical background in coding theory.

3 Remarks

In the above decoding procedure, we assumed that single errors get corrected after a few iterations. In fact if the noise value is ± 1, this happens in exactly one iteration. However, due to overlaps, we have to be careful with the effect of other neighboring clusters. For instance, it might be the case that two errors happen in a cluster and because it is not guaranteed to correct two errors, the cluster diverges from the original pattern and introduces even more errors. Therefore, we have to make sure that as long as the number of initial errors is less than a threshold, error correction operation does not increase the number of erroneous nodes.

4 Analysis of the Algorithm

We can think about analyzing the error correction performance of the network in different ways. One way is to perform an analysis similar to density evolution. The other way is to analyze in a similar to the peeling decoder [1]. However, in both cases, there is important differences between our error correcting algorithm and those in belief propagation and peeling decoder:

1. The main difference between the peeling decoder and the one we propose here is that in contrast to the peeling algorithm, a non-corrupted node might become corrupted with a small probablity in each iteration in our algorithm.

2. Even more importantly, the amount of error in each node is different, which is in contrast to density evolution for binary channels. Here, in order to track the probability of error, we must also track the number of nodes with different noise amplitudes.

At the moment, we have several ideas about possible ways of analysis. We propose one of them here. The others are subject to future investigations.

In what follows, we assume the bit-flipping error correction algorithm in [2] is being used, i.e. each pattern neurons updates its value only if:

1. The *number of* received feedbacks from its neighbors normalized by its out-degree is larger than a threshold φ.

2. In that case the node is updated according to the sign of the net input sum.

4.1 Notations

Let $\mathcal{E}_t$ denote the set of erroneous pattern nodes at iteration t, and $\mathcal{N}(\mathcal{E}_t)$ be the set of constraint nodes that are connected to the nodes in $\mathcal{E}_t$, i.e. these are the constraint nodes that have at least one neighbor in $\mathcal{E}_t$. In addition, let $\mathcal{N}^c(\mathcal{E}_t)$ denote the other constraint neurons that do not have any connection to any node in $\mathcal{E}_t$. Denote also the average neighborhood size of $\mathcal{E}_t$ by $S_t = \mathbb{E}(|\mathcal{N}(\mathcal{E}_t)|)$. In appendix (A) we will show that on average

$$S_t = m\left(1 - (1 - \frac{\bar{d}}{m})^{|\mathcal{E}_t|}\right) \tag{1}$$

where $\bar{d}$ is the average degree for pattern nodes.

To track the number of noisy nodes with different amplitudes, let $\mathcal{E}_t^{(i)}$ be the set of nodes that are corrupted by a $\pm i$ noise. Let $e_i(t) = |\mathcal{E}_t^{(i)}|$ be the number of such nodes. Therefore $e(t) = |\mathcal{E}_t| = \sum_{i\geq 1} e_i(t)$. Finally, Let $\mathcal{C}_t$ be the set of correct pattern nodes.

4.2 Recursive Relationship on The Number of Noisy Nodes

It is easy to see that the number of noisy nodes may change due to two events:

Event 1 . A node $x \in \mathcal{C}_t$ gets updated in iteration $t+1$, and so become corrupted. In Let $P_1(t)$ denote the probability of this event, given that $x \in \mathcal{C}_t$.

Event 2 . A node $x \in \mathcal{E}_t^{(1)}$ gets corrected in iteration $t+1$. Let $P_2(t)$ denote the probability of a noisy node gets updated towards the correct direction, given that $x \in \mathcal{E}_t$.

In the first(second) event above, the number of noisy nodes, $e(t)$, increases(decreases) by one in iteration $t+1$. Therefore, since all nodes update themselves independently and locally, one can write:

$$\begin{aligned} e(t+1) &= e(t) + n\left[\Pr\{x \in \mathcal{C}_t, \text{x gets updated}\} - \Pr\{x \in \mathcal{E}_t^{(1)}, \text{x gets corrected}\}\right] \\ &= e(t) + n\left[\Pr\{x \in \mathcal{C}_t\}P_1(t) - \Pr\{x \in \mathcal{E}^{(1)}\}P_2(t)\right] \\ &= e(t) + n\left[\left(1 - \frac{e(t)}{n}\right)P_1(t) - \left(\frac{e_1(t)}{n}\right)P_2(t)\right] \\ &= e(t) + \left[(n - e(t))\,P_1(t) - (e_1(t))\,P_2(t)\right] \end{aligned} \tag{2}$$

4.3 Recursive Relationship on $e_i(t)$

Similar to the above reasoning, one can write the set of recursive equations for $e_i(t)$. For $i > 1$ we have:

$$\begin{aligned} e_i(t+1) &= e_i(t) + n\left[\Pr\{x \in \mathcal{E}_t^{(i+1)}, \text{x gets corrected}\} + \Pr\{x \in \mathcal{E}_t^{(i-1)}, \text{x doesn't get corrected}\} - \Pr\{x \in \mathcal{E}_t^{(i)}\}\right] \\ &= e_i(t) + n\left[\Pr\{x \in \mathcal{E}_t^{(i+1)}\}P_2(t) + \Pr\{x \in \mathcal{E}_t^{(i-1)}\}(1 - P_2(t)) - \Pr\{x \in \mathcal{E}_t^{(i)}\}\right] \\ &= e_i(t) + n\left[\frac{e_{i+1}(t)}{n}P_2(t) + \frac{e_{i-1}(t)}{n}(1 - P_2(t)) - \frac{e_i(t)}{n}\right] \\ &= \frac{e_{i+1}(t)}{n}P_2(t) + \frac{e_{i-1}(t)}{n}(1 - P_2(t)) \end{aligned} \tag{3}$$

For $i = 1$, one has to include the effect of non-corrupted nodes in iteration t becoming noisy in iteration $t+1$:

$$\begin{aligned} e_1(t+1) &= e_1(t) + n\left[\Pr\{x \in \mathcal{E}_t^{(2)}, \text{x gets corrected}\} + \Pr\{x \in \mathcal{C}_t, \text{x gets updated}\} - \Pr\{x \in \mathcal{E}_t^{(1)}\}\right] \\ &= e_1(t) + n\left[\Pr\{x \in \mathcal{E}_t^{(2)}\}P_2(t) + \Pr\{x \in \mathcal{C}_t\}(P_1(t)) - \Pr\{x \in \mathcal{E}_t^{(1)}\}\right] \\ &= e_1(t) + n\left[\frac{e_2(t)}{n}P_2(t) + (1 - \frac{e(t)}{n})P_1(t) - \frac{e_1(t)}{n}\right] \\ &= \frac{e_2(t)}{n}P_2(t) + (1 - \frac{e(t)}{n}P_1(t) \end{aligned} \tag{4}$$

4.4 Deriving $P_1(t)$

$P_1(t)$ is the probability that a correct node gets updated in iteration t and becomes corrupted. Let P_1^x be the probability that a node $x \in \mathcal{C}_t$ with degree d_x updates its state. We have:

$$P_1^x = \Pr\{\frac{|\mathcal{N}(\mathcal{E}_t) \cap \mathcal{N}(x)|}{d_x} \geq \varphi\} \tag{5}$$

where $\mathcal{N}(x)$ is the neighborhood of x. Assuming random construction of the graph and relatively large graph sizes, one can approximate P_1^x by

$$P_1^x = \sum_{i=\lceil \varphi d_x \rceil}^{d_x} \binom{d_x}{i} \left(\frac{S_t}{m}\right)^i \left(1 - \frac{S_t}{m}\right)^{d_x - i}$$

$$= \sum_{i=\lceil \varphi d_x \rceil}^{d_x} \binom{d_x}{i} \left(1-(1-\frac{\bar{d}}{m})^{e(t)}\right)^i \left(1-\frac{\bar{d}}{m}\right)^{e(t)(d_x-i)} \quad (6)$$

where we have pluged in the result of the equation (1).

As a result of the above equations, we have:

$$P_1 = \mathbb{E}_{d_x}(P_1^x) \quad (7)$$

In case of the winner take-all algorithm where $\varphi = 1$, the above equations simplify to:

$$P_1^x = \left(\frac{S_t}{m}\right)_x^d = \left(1-(1-\frac{\bar{d}}{m})^{e(t)}\right)_x^d \quad (8)$$

And therefore

$$P_1 = \mathbb{E}_{d_x}(P_1^x) = \lambda(\frac{S_t}{m}) \quad (9)$$

where $\lambda = \sum_i \lambda_i x^i$ is the pattern nodes degree distribution polynomial.

4.5 Deriving $P_2(t)$

A node $x \in \mathcal{E}_t$ makes a correct decision if the net input sum it receives has the same sign as the sign of noise it experiences. Instead of finding an exact relation, we bound this probability by the probability that the neuron x shares less than half of its neighbors with other neurons, i.e. $P_2 \geq \Pr\{\frac{|\mathcal{N}(\mathcal{E}^*{}_t)\cap\mathcal{N}(x)|}{d_x} < 1/2\}$, where $\mathcal{E}^*{}_t = \mathcal{E}_t \setminus x$. Letting $P_2^x = \Pr\{\frac{|\mathcal{N}(\mathcal{E}^*{}_t)\cap\mathcal{N}(x)|}{d_x} < 1/2 | \deg(x) = d_x\}$, we will have:

$$\begin{aligned} P_2^x &= \sum_{i=0}^{\lfloor d_x/2 \rfloor} \binom{d_x}{i} \left(\frac{S_t^*}{m}\right)^i \left(1-\frac{S_t^*}{m}\right)^{d_x-i} \\ &= \sum_{i=0}^{\lfloor d_x/2 \rfloor} \binom{d_x}{i} \left(1-(1-\frac{\bar{d}}{m})^{e(t)-1}\right)^i \left(1-\frac{\bar{d}}{m}\right)^{(e(t)-1)(d_x-i)} \end{aligned} \quad (10)$$

where $S_t^* = \mathbb{E}(|\mathcal{N}(\mathcal{E}_t^*)|)$

Therefore, we will have:

$$P_2 \geq \mathbb{E}_{d_x}(P_2^x) \quad (11)$$

4.6 Wrapping Up

In order to show that the error correction algorithm corrects the initial number of noisy nodes, we have to show that $e(t) \to 0$ as $t \to \infty$. One approach is to write equations (2), (3) and (4) in terms of differential equations, find their closed form answer and derive the conditions under which the stable state is $e(t) = 0$.

The other possible approach is using equation (2) to show that $e(t+1) - e(t) < 0$, i.e. $(n - e(t))\, P_1(t) - (e_1(t))\, P_2(t) < 0$. Although the exact analysis remains a challenge, an interesting phenomenon has been observed in numerical analysis: If in the first iteration we have $(n - e(0))\, P_1(0) < e_1(0) P_2(0)$, then the algorithm converges and diverges otherwise. This condition is much easier to satisfy since in the first round we know the exact value of $e(0)$ and $e_1(0)$. Furthermore, since $P_1(0) = f(e(0))$ and $P_2(0) \geq g(e(0))$, one can find a condition on $e(0)$, the initial number of noisy nodes, uder which the algorithm converges. Finding an exact such relationship remains a open issue at the moment.

5 Summary of the paper: Efficient Erasure Correcting Codes [1]

In [1] the authors introduce a method to efficiently correct erasures in a Binary Erasure Channel (BEC). The main idea of the paper is to cascade several randomly-constructed graphs to form a nested code. The first graph has k variable nodes and βk redundancy bits, i.e. these βk nodes enforce constraints on the k variable nodes. The second stage has βk variable nodes and $\beta^2 k$ redundancy bits, i.e. the second β^k redundancy bits enforce some constraint on the first βk redundancy bits. This process continues m times such that $\beta^m k \simeq \sqrt{k}$. In the last stage, a traditional erasure correcting code is applied.

In the decoding process, the decoder starts from the last stage and recovers remaining erasures in each stage. The main idea in the analysis of the decoding algorithm is based on tracking the fraction of degree-one check nodes over the residual error correcting graph. As long as this fraction is positive, the algorithm can proceed. Therefore, in order to have a good code, we must find degree distributions for which the fraction of degree-one nodes remain positive during the course of algorithm.

To analyze the decoding process, the authors restrict themselves to a sub-graph composed of the erased nodes that have not be corrected so far and those check nodes connected to them. Then, the decoding process corresponds to finding a check node with degree one over this residual graph, correct the connected message node to this node, send the corrected message to all its neighbors and remove the corrected message node as well as all its edges.

As a result, as long as the fraction of degree one check nodes are positive, the algorithm can progress. This is a key idea in analyzing the decoding process. To prove the success of the decoding algorithm, one has to also show that the final stages of the algorithm corrects errors, i.e. these are the stages that asymptotic arguments do not work since there are only a few edges are remaining in the residual graph. The key technique here is to show that the random graph remaining at this stage has good expansion properties.

The authors analyze the average behavior of the algorithm but in *Theorem 1* provide a concentration argument showing that any instance of the random ensemble do not deviate too much from the average behavior.

6 Future Works

In the upcoming weeks, we are going to try different ways of analyzing the error correcting algorithms. We have two approaches in mind at the moment:

- Model a nods with noise values $\pm i$ by i copies of the same node and assume each copy has a noise of ± 1. This way, one can only with degree distributions and how it evolves, very similarly to the one presented in [1].
- Extend density evolution approaches over non-binary channels to the model at hand.

We will bring the result of our progress in the next report.

A Average neighborhood size

In this appendix, we find an expression for the average neighborhood size for erroneous node, $S_t = \mathbb{E}(|\mathcal{N}(\mathcal{E}_t)|)$. To this end, we assume the following procedure for constructing a right-irregular

bipartite graph:

- In each iteration, we pick a variable node x with a degree randomly determined according to the given degree distribution.
- Based on the given degree d_x, we pick d_x constraint nodes uniformly at random *with replacement* and connect x to the constraint node.
- We repeat this process n times, until all variable nodes are connected.

Note that the assumption that we do the process with replacement is made to simplify the analysis. This assumption becomes more exact as n grows.

Having the above procedure in mind, we will find an expression for the average number of constraint nodes in each construction round. More specifically, we will find the average number of constraint nodes connected to i pattern nodes at round i of construction. This relationship will in turn yields the average neighborhood size of $|\mathcal{E}_t|$ erroneous nodes in iteration t of error correction algorithm.

With some abuse of notations, let S_e denote the number of constraint nodes connected to pattern nodes in round e of construction procedure mentioned above. We write S_e recursively in terms of e as follows:

$$\begin{aligned} S_{e+1} &= \mathbb{E}_{d_x}\{\sum_{j=0}^{d_x}\binom{d_x}{j}\left(\frac{S_e}{m}\right)^{d_x-j}\left(1-\frac{S_e}{m}\right)^{j}(S_e+j)\} \\ &= \mathbb{E}_{d_x}\{S_e + d_x(1-S_e/m)\} \\ &= S_e + \bar{d}(1-S_e/m) \end{aligned} \tag{12}$$

Where $\bar{d} = \mathbb{E}_{d_x}\{d_x\}$ is the average degree of the pattern nodes. In words, the first line calculates the average growth of the neighborhood when a new variable node is added to the graph. The proceeding equalities directly follows from relationship on binomial sums. Noting that $S_1 = \bar{d}$, one obtains:

$$S_t = m\left(1-(1-\frac{\bar{d}}{m})^{|\mathcal{E}_t|}\right) \tag{13}$$

In order to verify the correctness of the above analysis, we have performed some simulations for different network sizes and degree distributions obtained from the graphs returned by the learning algorithm. We generated 100 random graphs and calculated the average neighborhood size in each iteration over these graphs. Furthermore, two different network sizes were considered $n = 100, 200$ and $m = n/2$ in all cases, where n and m are the number of pattern and constraint nodes, respectively. The result for $n = 100, m = 50$ is shown in Figure 2, where the average neighborhood size in each iteration is illustrated and compared with theoretical estimations given by equation(13). Figure 3 shows similar results for $n = 200, m = 100$. In the figure,the dashed line shows the average neighborhood size over these graphs. The solid line corresponds to theoretical estimations. It is obvious that the theoretical value approximates the simulation results rather exactly.

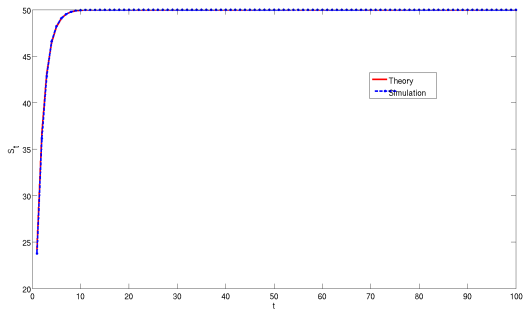

Figure 2: The theoretical estimation and simulation results for the neighborhood size of irregular graphs with a given degree-distribution for $n = 100$, $m = 50$ and over 2000 random graphs.

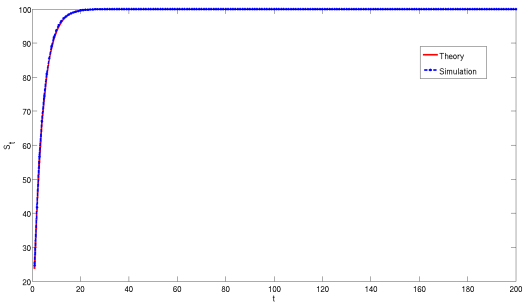

Figure 3: The theoretical estimation and simulation results for the neighborhood size of irregular graphs with a given degree-distribution for $n = 200$, $m = 100$ and over 2000 random graphs.

References

[1] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, *Efficient erasure coding codes*, IEEE Tran. Inf. Theory, Vol. 47, No. 2, 2001, pp. 569-584.

[2] A. H. Salavati, A. Karbasi, *Multi-Level Error-Resilient Neural Networks with Learning* , Submitted to the IEEE Int. Sym. Inf. Theory (ISIT 2012), 2012.

[3] K.R. Kumar, A.H. Salavati and A. Shokrollahi, *Exponential pattern retrieval capacity with non-binary learning associative memory*, Under preparation.

[4] K.R. Kumar, A.H. Salavati and A. Shokrollahi, *Exponential pattern retrieval capacity with non-binary associative memory*, Proc. IEEE Information Theory Workshop, 2011.