Progress Report
1-16 May 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)

June 5, 2012

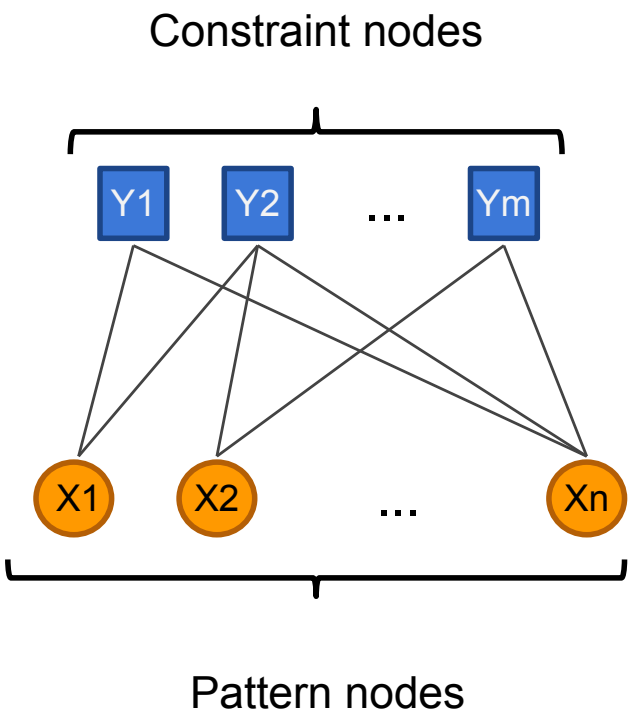


Figure 1: A bipartite graph that represents the constraints on the training set.

1 Summary

In the last two weeks, Mr. Amin Karbasi and I were busy finding an altrenative alnaysis for the proposed neural algorithm, based on the ideas Prof. Amin Shokrollahi had mentioned in our meeting. In what follows, you can find the list of ideas as well as the details of our investigation for some of them.

We also prepared the paper which we submitted to the NIPS 2012 conference. The paper is based on the idea of clustered neural associative memory and the error correction results are absolutely impressive compared to our previous approaches.

2 The Ideas to Analyze the Error Correcting Algorithm

Here, we assume that we are in the realm of single-cluster neural model, shown by figure 1, unless specified otherwise.

Now we would like to analyze the behavior of the error correcting Algorithm 1. There are several ideas at the moment to investigate:

1. Avoiding decoding errors
2. Stability condition in the first iteration
3. Polynomial evolution

Algorithm 1 Recall Algorithm: Bit-Flipping

Input: Connectivity matrix W, threshold φ, iteration $t_{\max}$

Output: $x_1, x_2, \ldots, x_n$

1: **for** $t = 1 \to t_{\max}$ **do**

2: *Forward iteration:* Calculate the weighted input sum $h_i = \sum_{j=1}^{n} W_{ij}^b x_j$, for each neuron y_i and set:

$$y_i = \begin{cases} 1, & h_i < 0 \\ 0, & h_i = 0 \\ -1, & \text{otherwise}^2 \end{cases}$$

3: *Backward iteration:* Each neuron x_j with degree d_j computes

$$g_j^{(1)} = \frac{\sum_{i=1}^{m} W_{ij}^b y_i}{d_j}, g_j^{(2)} = \frac{\sum_{i=1}^{m} |W_{ij}^b y_i|}{d_j}$$

4: Update the state of each pattern neuron j according to $x_j = x_j + \text{sign}(g_j^1)$ only if $|g_j^{(2)}| > \varphi$.

5: $t \leftarrow t + 1$

6: **end for**

4. Density Evolution
5. Product codes
6. Generalized Tanner graphs

Each idea is discussed in more details below.

2.1 Avoiding decoding errors

This would be the safe road towards analyzing the algorithm. In this approach, we are going to modify the decoding algorithm in such a way that it runs for one iteration. If it was able to correct the input noise completely, we maintain the new state of neurons as the correct state. Otherwise, if there were some constraints still violated, we ignore this iteration and keep the last value of neurons. This way, we are sure that the decoding operation does not introduce any new errors.

Although it might be sub-optimal, this approach will help us analyze the clustered version of the model in which each cluster makes sure to correct one error and not more. Then, we can apply the analysis suggested in [1] to analyze the algorithm.

2.2 Stability condition in the first iteration

As derived in our previous two reports, the probability of making a decoding error and correcting a node is a function of the error probability. We would like to track this probability and obtain conditions under which it evolves to zero.

However, our numerical analysis shows that if the error probability does not decrease in the first iteration, then it is going to diverge to 1. On the other hand, if we manage to decrease the error probability in the first round, the probability of error converges to zero. Therefore, for a successful

decoding (in the 1-cluster network) all we need to do is to focus on the first round (although there is no mathematical proof yet).

2.3 Polynomial evolution

Amin suggested that we formulate the probability of different noise values into a polynomial and track the evolution of this polynomial. More specifically, in the suggested approaches in previous reports, we denoted the probability of a node having $\pm i$ noise by $p_i(t)$. Then, we found a recursive equation for $p_i(t)$ based on $p_{i+1}(t)$ and $p_{i-1}(t)$. Now, let us define $\Psi(x,t) = \sum_i p_i(t)x^i$. Then, we might be able to write $\Psi(x,t+1)$ as a function of $\Psi(x,t)$. In the end, we would like the coefficients of this polynomial all go to zero except for the $p_0(t)$.

2.4 Density evolution

If we assume we are going to recall the all-zero pattern, then the messages transmitted by pattern nodes are pure noise. We would like to track the pdf of these messages and show that it converges to $\delta(0)$. Similar to the density evolution technique in codes on graphs, we might be able to formulate our problem in this framework. The only thing we need to do is to write the two equations according to which the density evolves at pattern and constraint nodes.

This technique is very useful in analyzing a variant of the decoding algorithm in which symmetrical weights are used.

2.5 Product codes

It would be possible to analyze the clustered version of the code if we consider a 2-D clustering technique, i.e. put nodes inside each cluster in rows of a matrix, then enforce another code (by clustering) over columns. In this way, each row and column of this matrix is capable of correcting one error and decoding failure corresponds to finding a 2×2 clique in this random graph.

However, this scheme also suffers from not considering the probability of errors introduced during the decoding operation.

2.6 Generalized Tanner graphs

We can model each cluster by generalized check node in a generalized Tanner graph and use belief propagation technique. This is actually something we have considered before and it also suffers from not considering the probability of errors introduced during the decoding operation.

Among the ideas above, we will investigate two in more details: the first one, which is our lifeline, was eventually used in the NIPS 2012 paper and more details can be found there. We explore the second idea below. It is more advanced than the previous one and as expected more difficult. The third one looks very promising as well and in future we may consider that as well.

3 Error Analysis: Stability Condition

One of the ideas that seems promising is to ensure progress in error correction in the first round. Although not mathematically proven yet, our numerical analysis shows that if the algorithm fails to reduce the probability of error in the first round, it will get stuck. In what follows, we first

go over the mathematical relationship describing the evolution of error. Next, we will derive the condition under which progress in the first round is ensured. We finish the analysis by reporting some numerical results which confirm our findings.

3.1 Notations and Model

In this approach, without loss of generality, we assume that the pattern we would like to memorize is the all-zero pattern. Therefore, the non-zero messages transmitted by the pattern nodes are pure noise. Let $z(t) \in \mathcal{S}^n$ denote the transmitted message by the pattern nodes, where $\mathcal{S} = \{-S, \dots, S\}$ is the noise alphabet. At constraint nodes, we have $y(t) = \text{sign}(Wz(t))$.

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 $e(t) = |\mathcal{E}_t| = \sum_{i \geq 1} e_i(t)$. Furthermore, let $p_i(t)$ be the *average* probability that a given pattern node x belongs to $\mathcal{E}_t^{(i)}$.

Let $q_w^e(t)$ and $q_c^e(t)$ be the probability of a wrong and correct update by a node x in the set of noisy nodes, respectively. Furthermore, let $q_w^c(t)$ be the probability of a (wrong) update for node x in the set of non-corrupted nodes. We will write recursive equations for $p_i(t)$ and try to formulate the density evolution of the proposed error correction algorithm.

3.2 Error Evolution

In order to analyze the algorithm, we write the recursive equations for $p_i(t)$ considering the behavior of a single node x. For $1 < i < S$ we have:

$$p_i(t+1) \quad = \quad p_{i+1}(t) q_c^e(t) + p_{i-1}(t) q_w^e(t) \tag{1}$$

and for $i = 1$ we have:

$$p_1(t+1) \quad = \quad p_2(t) q_c^e(t) + p_0(t) q_w^c(t) \tag{2}$$

Finally, for $i = S$ we have:

$$p_S(t+1) \quad = \quad p_{S-1}(t) q_w^c(t) + p_S(t) q_w^e(t) \tag{3}$$

Obviously, the probabilities $q_c^e(t)$, $q_w^e(t)$ and $q_w^c(t)$ depend on $p_i(t)$'s as well. Therefore, we must derive their dependency. Let $q(t)$ be the probability of a pattern node being noisy. Furthermore, let π_1^y be the probability of a constraint node y reporting a constraint violation, assuming its degree is i. Finally, denote the average probability of a constraint violation by π^y.

Since we assume the probability of noise terms canceling out in the constraint nodes is negligible, the probability that a constraint node send a value other than zero, i.e. reporting a constraint violation, would be the probability that at least one of its neighbors is noisy. Thus

$$\pi_1^y(t) = 1 - (1 - q(t))^i \tag{4}$$

And therefore

$$\pi^y(t) = \mathbb{E}_i\{1 - (1 - q(t))^i\} = 1 - (1 - q(t))\rho(1 - q(t)) \tag{5}$$

where $\rho(x) = \sum \rho_i x^{i-1}$.

3.2.1 Deriving $q_w^c(t)$

Now in order to derive $q_w^c(t)$, note that a node $x \in \mathcal{C}_t$ gets updated in iteration t only if it receives some feedback over at least a fraction φ of its input links. Let $q_w^{c_x}$ denote this probability, assuming the degree of node x is d_x. Then we have

$$q_w^{c_x}(t) = \sum_{i=\varphi d_x}^{d_x} \binom{d_x}{i} \left(\pi^y(t)\right)^i \left(1-\pi^y(t)\right)^{d_x-i} \tag{6}$$

Therefore,

$$q_w^c(t) = \mathbb{E}_{d_x}\{\varrho_1^x(t)\} = \sum_{d_x} \lambda_{d_x} \sum_{i=\varphi d_x}^{d_x} \binom{d_x}{i} \left(\pi^y(t)\right)^i \left(1-\pi^y(t)\right)^{d_x-i} \tag{7}$$

Assuming $\varphi = 1$, as in many practical cases, equations (6) and (7) simplify to

$$q_w^{c_x}(t) = \left(\pi^y(t)\right)^{d_x} = \left(1-(1-q(t))\rho(1-q(t))\right)^{d_x} \tag{8}$$

and

$$q_w^c(t) = (1-(1-q(t))\rho(1-q(t)))\lambda(1-(1-q(t))\rho(1-q(t))) \tag{9}$$

where $\lambda(x) = \sum \lambda_i x^{i-1}$.

3.2.2 Deriving $q_c^e(t)$

In order to derive the probability of a noisy node makes a correct decision, we note that if a constraint node is only connected to this node, then it sends the correct sign of noise with probability 1. However, if the same constraint node is also connected to other noisy pattern node(s), the message it sends would depend on the sign and amount of noise from the other noisy node(s). We assume in this case the noisy node sends $+1$ and -1 each with probability $1/2$ (Let's call this assumption *A1*).[1] Therefore, a noisy pattern node receives a correct update message from a neighboring constraint node with probability 1 if it is not connected to any other noisy node(s), and with probability $1/2$ otherwise.

The probability that a constraint node is not connected to any other noisy node is the probability that it receives correct messages over its other links. Therefore, this probability would be $\mathbb{E}_{d_y}\{(1-q(t))^{d_y-1}\} = \rho(1-q(t))$.

As a result, a noisy pattern node x with degree d_x makes a correct decision if the number of correct feedbacks it receives exceeds the incorrect ones. Mathematically speaking

$$\begin{aligned} q_c^{e_x}(t) &= \sum_{i=\lceil d_x/2 \rceil}^{d_x} \binom{d_x}{i} \left(\rho(1-q(t)) + .5(1-\rho(1-q(t)))\right)^i \left(.5(1-\rho(1-q(t)))\right)^{d_x-i} \\ &= (1/2)^{d_x} \sum_{i=\lceil d_x/2 \rceil}^{d_x} \binom{d_x}{i} \left(1+\rho(1-q(t))\right)^i \left(1-\rho(1-q(t))\right)^{d_x-i} \end{aligned} \tag{10}$$

[1] This assumption seems *intuitively* correct but needs further justification.

And therefore

$$\begin{aligned} q_c^e(t) &= \mathbb{E}\{q_c^{e_x}(t)\} \\ &= \sum_{d_x} \lambda_{d_x} (1/2)^{d_x} \sum_{i=\lceil d_x/2 \rceil}^{d_x} \binom{d_x}{i} \left(1+\rho(1-q(t))\right)^i \left(1-\rho(1-q(t))\right)^{d_x-i} \end{aligned} \tag{11}$$

3.2.3 Wrapping up with deriving $q(t)$

Finally, we have to find a recursive equation for the probability of a node being in noisy state in iteration t. We have:

$$\begin{aligned} q(t+1) &= q(t) + \Pr\{x \in \mathcal{C}_t\} q_w^c(t) - \Pr\{x \in \mathcal{E}_t^{(1)}\} q_c^e(t) \\ &= q(t) + (1-q(t)) q_w^c(t) - p_1(t) q_c^e(t) \end{aligned} \tag{12}$$

4 Stability Condition

Now we would like to see under what circumstances $q(1) < q(0)$, where $q(0)$ is the error probability of the environment. Using equation (12), we would like to have

$$(1-q(0)) q_w^c(0) < p_1(0) q_c^e(0) \tag{13}$$

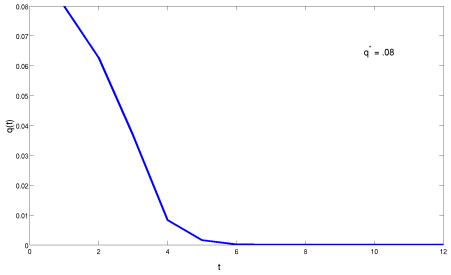
where $q_w^c(0)$ ad $q_c^e(0)$ can be replaced with the results of equations (7) and (11). Although we can not solve the given equations analytically, except may be in some special cases, numerical analysis could help us find the maximum $q(0)$ which satisfies this condition for given degree distributions $\lambda(x)$ and $\rho(x)$. We claim that this threshold, denoted by q^* is the threshold which determines not only the behavior of the algorithm in the first iteration, but also in the course of the algorithm. Therefore, if $q(0) \leq q^*$, Algorithm 1 should converge by correcting all errors (with vanishing *bit* error probability). However, if $q(0) > q^*$, the algorithm diverges and $q(t) \to 1$ as $t \to \infty$.

5 Numerical Analysis Results

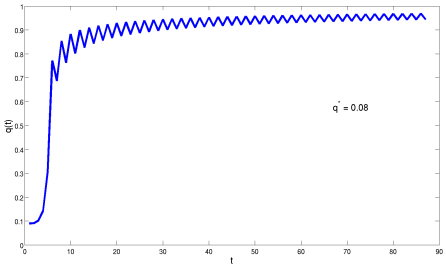
To verify the above claim, we have deigned a process to emulate the dynamic behavior of the algorithm, according to equations (1), (2) and (3). We consider two different settings: random ensembles and learned ensembles. In the first case, we have $n = 200$, $k = 100$ and we generate pairs of degree distributions $\lambda(x)$ and $\rho(x)$ at random, each according to a given distribution (we consider regular, Gaussian and Poisson distributions). We then find the threshold q^* from the inequality (13). Finally, in order to verify the correctness of threshold, we initialize $p_1(0) = q^*$, $p_i(0) = 0$ for $i > 1$ and let the probability distribution evolve according to equations (1), (2) and (3). S is fixed to 20. After the algorithm converged, we look at the probability distribution and if we had $p_i = 0, \forall i > 0$, it means $q(t) \to 0$, i.e. the algorithm has been successful. Otherwise, we declare failure. We count the number of successful attempts over 20 trials and denote the success rate by $\kappa(q^*)$. We repeat this operation with $p_1(0) = q^* + 0.01 > q^*$, i.e. decoding over the threshold and denote the success rate by $\kappa(q^* + 0.01)$. We expect to see $\kappa(q^*) \simeq 1$ and $\kappa(q^* + 0.01) \simeq 0$.

In the second setting, we repeat the same steps as above except with degree distributions taken from actual neural graphs learned by our learning algorithm.

In what follows, we will report the results of the numerical analysis for the above settings.



(a) $q(0) = q^*$



(b) $q(0) = q^* + 0.01$

Figure 2: The evolution of error probability $q(t)$ as a function of iteration t with $d_r = 10, d_c = 5$ and different $q(0)$.

d_c	d_r	q^*	$\kappa(q^*)$	$\kappa(q^* + 0.01)$
5	10	0.08	1.00	0
10	20	0.06	1.00	0
50	100	0.02	1.00	0

Table 1: The percentage of successful decoding above and below threshold, for different degrees of a regular bipartite graph.

5.1 Regular Ensembles

Figure 2 illustrates the behavior of $q(t)$ as a function of iteration number t for a regular ensemble with row degree $d_r = 10$ and column degree $d_c = 5$. As obvious from the figure, the error probability converges to zero if $q(0) \leq q^*$ and converges to 1 otherwise.

Table 1 summarizes the success rate, i.e. the cases for which $q(t) \to 0$, for different row and column degrees for the egular-bipartite graph. Again, the success rate is close to 1 for $q(0) \leq q^*$ and is close to 0 otherwise

5.2 Gaussian Degree Distribution

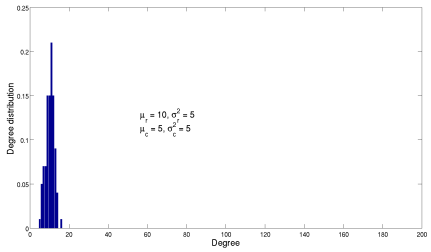
In this section, we consider Gaussian ensemble with row degree d_r $\mathcal{N}(\mu_r, \sigma_r^2)$ and column degree d_c $\mathcal{N}(\mu_r, \sigma_r^2)$. Figure 3 illustrates the behavior of $q(t)$ as a function of iteration number t for a Gaussian ensemble with $\mu_r = 10$, $\mu_c = 5$ and $\sigma_r^2 = \sigma_c^2 = 5$. In the same picture, degree distributions ar depicted as well. As obvious from the figure, the error probability converges to zero if $q(0) \leq q^*$ and converges to 1 otherwise.

Figure 4 illustrates the same behavior behavior as a function of iteration number t for $\mu_r = 40$, $\mu_c = 20$ and $\sigma_r^2 = \sigma_c^2 = 25$.

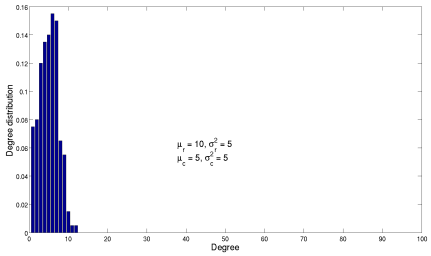
Table 2 summarizes the success rate for different Gaussian ensembles. Again, the success rate is close to 1 for $q(0) \leq q^*$ and is close to 0 otherwise

5.3 Poisson Degree Distrbution

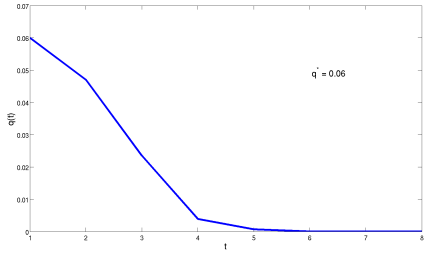
In this section, we consider Poisson ensemble with row degree d_r and column degree d_c distributed according to a Poisson distribution with parameters λ_r and λ_c, respectively.



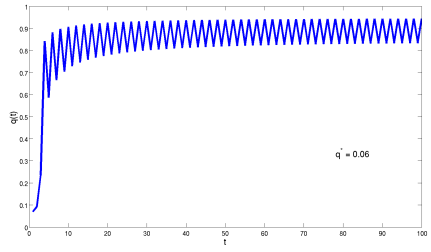
(a) Row degree distribution



(b) Column degree distribution

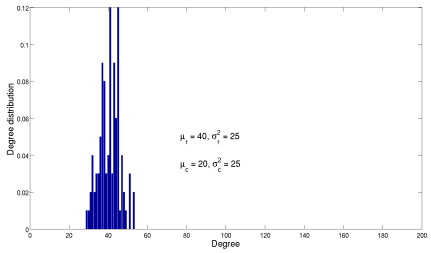


(c) $q(0) = q^*$

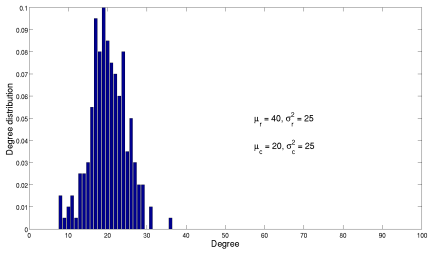


(d) $q(0) = q^* + 0.01$

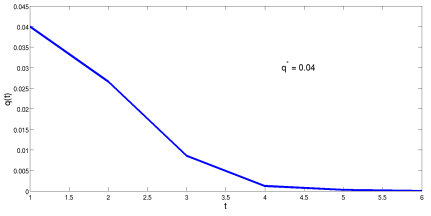
Figure 3: The evolution of error probability $q(t)$ as a function of iteration t for a Gaussian ensemble with $\mu_r = 10$, $\mu_c = 5$ and $\sigma_r^2 = \sigma_c^2 = 5$.



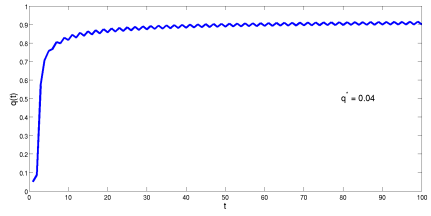
(a) Row degree distribution



(b) [Column degree distribution



(c) $q(0) = q^*$

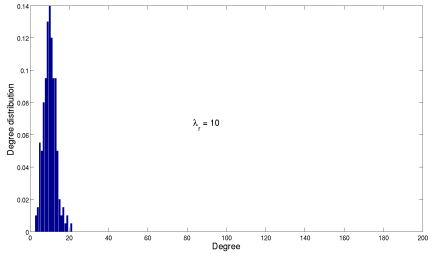


(d) $q(0) = q^* + 0.01$

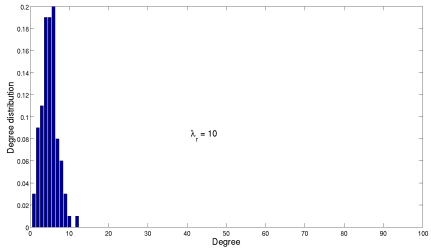
Figure 4: The evolution of error probability $q(t)$ as a function of iteration t for a Gaussian ensemble with $\mu_r = 40$, $\mu_c = 20$ and $\sigma_r^2 = \sigma_c^2 = 25$.

μ_c	μ_r	σ_c^2	σ_r^2	q^*	$\kappa(q^*)$	$\kappa(q^*+0.01)$
5	10	5	5	0.06	1.00	0
10	20	5	5	0.06	1.00	0
40	20	5	5	0.04	1.00	0
40	20	15	15	0.04	1.00	0

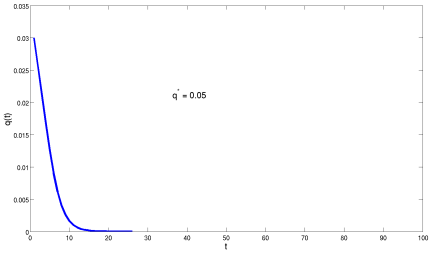
Table 2: The percentage of successful decoding above and below threshold, for different Gaussian ensembles.



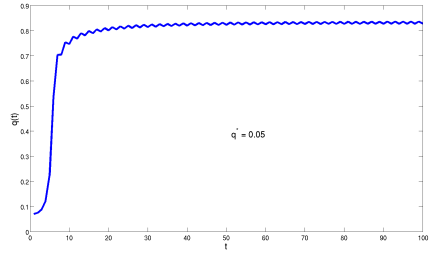
(a) Row degree distribution



(b) Column degree distribution



(c) $q(0) = q^*$



(d) $q(0) = q^* + 0.01$

Figure 5: The evolution of error probability $q(t)$ as a function of iteration t for a Poisson ensemble with $\lambda_r = 10$ and $\lambda_c = 5$.

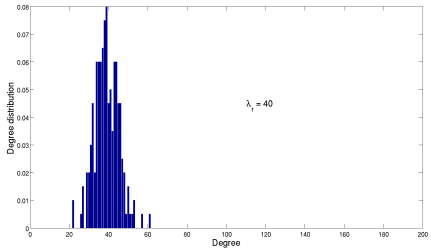
Figure 5 illustrates the behavior of $q(t)$ as a function of iteration number t for a Gaussian ensemble with $\mu_r = 10$, $\mu_c = 5$ and $\sigma_r^2 = \sigma_c^2 = 5$. In the same picture, degree distributions ar depicted as well. As obvious from the figure, the error probability converges to zero if $q(0) \leq q^*$ and converges to 1 otherwise.

Figure 6 illustrates the same behavior behavior as a function of iteration number t for $\mu_r = 40$, $\mu_c = 20$ and $\sigma_r^2 = \sigma_c^2 = 25$.

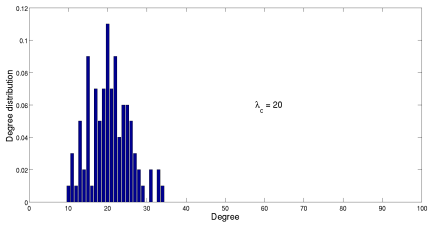
Table 3 summarizes the success rate for different Gaussian ensembles. Again, the success rate is close to 1 for $q(0) \leq q^*$ and is close to 0 otherwise

5.4 Learned Ensembles

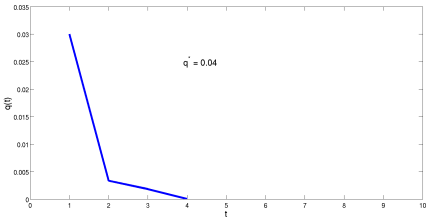
Now we switch our attention to the neural networks learned by the proposed learning algorithm in [3]. Figure 7 illustrates the behavior of $q(t)$ as a function of iteration number t for the considered neural network with learning parameters $\alpha = 0.75$, $\beta = 0.45$ and $\theta = 0.15$. In the same picture, degree distributions ar depicted as well. As obvious from the figure, the error probability converges



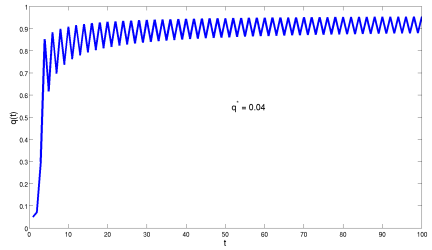
(a) Row degree distribution



(b) Column degree distribution



(c) $q(0) = q^*$

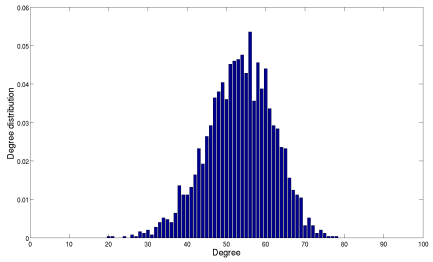


(d) $q(0) = q^* + 0.01$

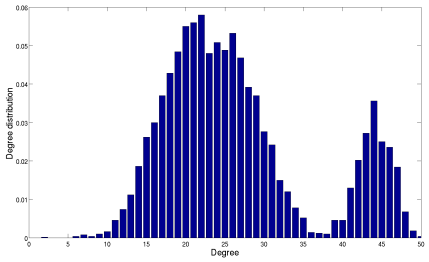
Figure 6: The evolution of error probability $q(t)$ as a function of iteration t for a Poisson ensemble with $\lambda_r = 40$ and $\lambda_c = 20$.

λ_c	λ_r	q^*	$\kappa(q^* - 0.01)$	$\kappa(q^* + 0.01)$
5	10	0.05	1.00	0
10	20	0.06	1.00	0
20	40	0.04	1.00	0

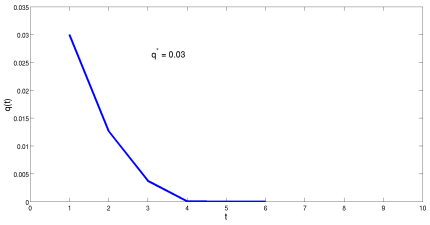
Table 3: The percentage of successful decoding above and below threshold, for different Poisson ensembles.



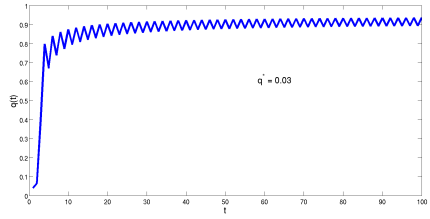
(a) Row degree distribution



(b) Column degree distribution



(c) $q(0) = q^*$



(d) $q(0) = q^* + 0.01$

Figure 7: The evolution of error probability $q(t)$ as a function of iteration t for a neural network learning parameters $\alpha = 0.75$, $\beta = 0.45$ and $\theta = 0.15$.

α	β	θ	q^*	$\kappa(q^*)$	$\kappa(q^* + 0.01)$
0.75	0.45	0.15	0.03	1.00	0
0.75	0.45	0.02	0.01	1.00	0
0.45	0.45	0.015	0.01	1.00	0

Table 4: The percentage of successful decoding above and below threshold, for different Poisson ensembles.

to zero if $q(0) \leq q^*$ and converges to 1 otherwise.

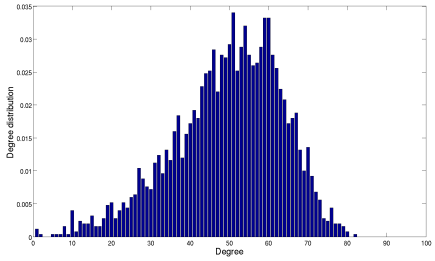
Figure 8 illustrates the same behavior for a neural network with learning parameters $\alpha = 0.75$, $\beta = 0.45$ and $\theta = 0.02$.

Table 4 summarizes the results of the numerical analysis for the learned neural graphs. Obviously, the results of the table confirm our approach.

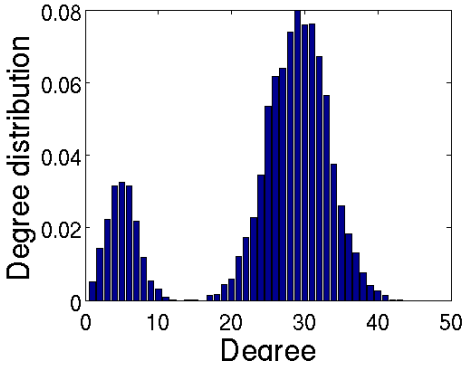
6 Some Observations

6.1 Very low and very high degrees are bad for performance

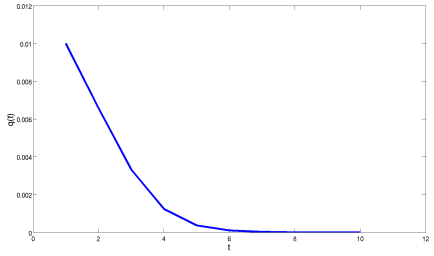
The simulation results show that when we have a combination of very low and high degrees, the error correction performance is poor. This in fact makes sense because we have this sort of combination, if the pattern node with very high degree is noisy, it results in the violation of a lot of constraint nodes. Then, there is a good chance that all of the neighbors of low degree pattern nodes lie within the set of violated constraints. As a result, these pattern nodes will be updated incorrectly by Algorithm 1. To avoid this phenomenon, we must make sure that all degrees are far from the two



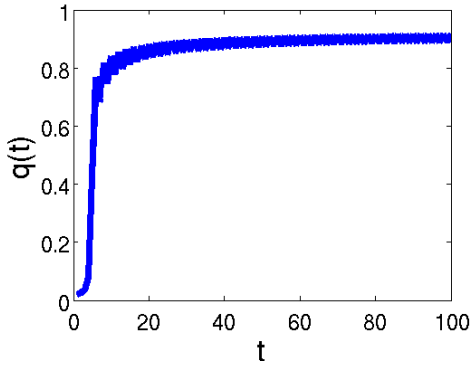
(a) Row degree distribution



(b) Column degree distribution

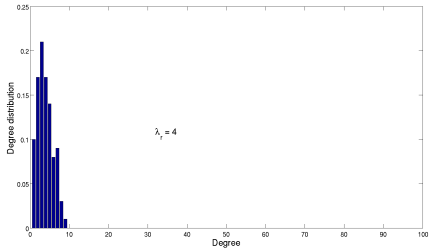


(c) $q(0) = q^*$

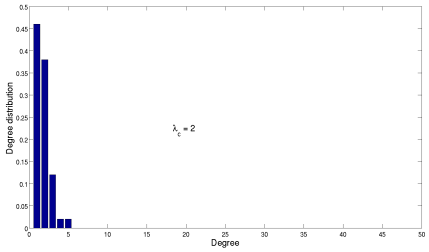


(d) $q(0) = q^* + 0.01$

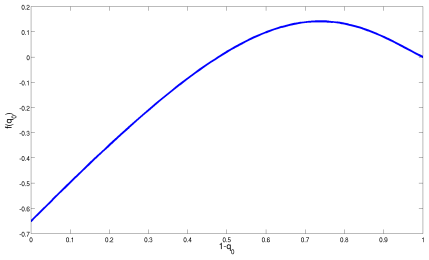
Figure 8: The evolution of error probability $q(t)$ as a function of iteration t for a neural network learning parameters $\alpha = 0.75$, $\beta = 0.45$ and $\theta = 0.02$.



(a) Row degree distributed according to the Poisson difstribution with parameter $\lambda_r = 4$



(b) Column degree distributed according to the Poisson difstribution with parameter $\lambda_c = 2$



(c) Evolution of the stability condition $f(q_0)$ as a function of $1 - q_0$

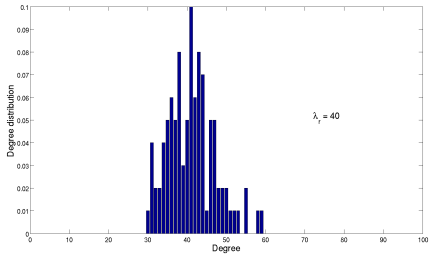
(d) The evolution of error probability $q(t)$ as a function of iteration t for a neural network learning parameters $\alpha = 0.75$, $\beta = 0.45$ and $\theta = 0.02$.

extremes of degree distribution.

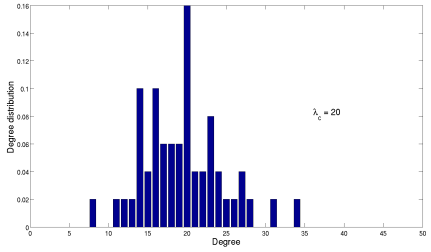
6.2 Ensembles without any decoding threshold

In some particular cases, it is possible that there are no solution q^* that satisfy inequality (13). In fact, there is a very close relationship between these ensembles and the observation above, i.e. in many cases, there are not threshold due to very low pattern node degrees. Figure 9d shows $f(q_0) = (1 - q(0))q_w^c(0) - p_1(0)q_c^e(0)$ as a function of $q(0)$ for a sample degree distribution, illustrated in the same graph. As a matter of fact, we are looking for the maximum $q(0)$ such that that $f(q_0) < 0$ (Note that the horizontal axis is $1 - q(0)$ so we should look for such q^* towards the right hand side of the horizontal axis). However, as can be clearly seen in the figure, $f(q_0) > 0, \forall q_0 = q(0)$. Therefore, the decoding operation would not have zero error probability under any circumstances (note that this does not mean the probability of error would be equal to 1 either).

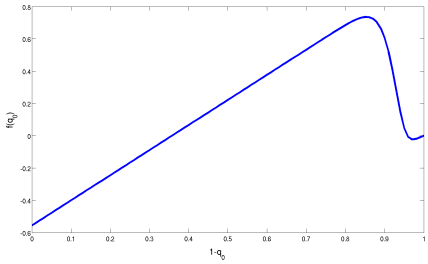
By carefully investigating $q_w^c(0)$ and $q_c^e(0)$, one can clearly see that the fraction of low degree nodes are partly responsible for this phenomenon. Figure 9h illustrates the same $f(q_0) = (1 - q(0))q_w^c(0) - p_1(0)q_c^e(0)$ but for a different degree distribution. It is clear that in this case $f(q_0) < 0$ for $q_0 = q(0) \leq q^* = 0.03$. Furthermore, by comparing the degree distribution in figure 9h and that of figure 9d one sees the effect of low degree pattern nodes on the existence threshold. The same phenomenon is also apparent when comparing Figure 7b and 8b. In the later case, the portion of low degree pattern nodes are higher which translates into lower decoding threshold ($q^* = 0.01$ versus $q^* = 0.03$ for degree distribution of Figure 8b).



(e) Row degree distributed according to the Poisson distribution with parameter $\lambda_r = 40$



(f) Column degree distributed according to the Poisson difstribution with parameter $\lambda_c = 20$



(g) Evolution of the stability condition $f(q_0)$ as a function of $1 - q_0$

(h) The evolution of error probability $q(t)$ as a function of iteration t for a neural network learning parameters $\alpha = 0.75$, $\beta = 0.45$ and $\theta = 0.02$.

7 Conclusions and Future Works

In this report, using numerical analysis we showed that in order to have successful decoding for a single-cluster network, as shown in Figure 1, it is sufficient to only ensure the success fo decoding in the first round. We formulated this requirement in terms of inequality (13) which we could use to obtain the maximum error probability for a given degree distribution that still ensures successful decoding.

At this point, there are two very important steps:

1. Extend the analysis to clustered networks, i.e. those that have multiple clusters.

2. Prove the sufficiency of condition (13) for the success of decoding process.

Regarding the first step, we should derive the conditions for the success of all clusters. We could use the results found in this report in order to ensure the decoding operation itself does not introduce more errors. From Figure 2a and other similar graphs, it is obvious that as long as the initial noise probability is less than a threshold, the overall probability of decoding error decreases with iteration. Therefore, we can use this condition to prove the convergence of decoding in each cluster. However, for those clusters that the initial probability of error is larger than this threshold, the block error probability reaches 1. Therefore, **for these clusters we must revert everything back to the initial values in case decoding fails**.

Step two above requires further investigations.

There is also another important points which we should consider in our future simulations. From observations made in section 6, it is obvious that both too small and too large degrees are

undesirable. Therefore, in our learning algorithm, we must tune the learning parameters such that the resulting neural graph satisfies these two requirements. In this case, we will get much better error performance. Our new simulation results (not shown in this report) confirm this conjecture.

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, A. Shokrollahi, *A Non-Binary Associative Memory with Exponential Pattern Retrieval Capacity and Iterative Learning*, To be submitted to IEEE Trans. Neuroscience