Progress Report
17-30 March 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)

April 2, 2012

1 Summary

In the last two weeks, I was mainly busy working on the journal paper. I have made some progress both in terms of proving the convergence of the learning algorithm as well as analyzing the probability of recall error for the proposed neural associative memory.

Regarding the multi-level neural associative memory, together with Mr. Amin Karbasi, we worked on finding a proper model in order to be able to analyze the algorithm we submitted to ISIT 2012. We are going to first do a sanity check on the neural aspect of the model with the help of experts on the neural networks. Having done that we will work on analyzing the performance of the proposed method. I also worked on the MATLAB code for the adaptive single-level neural associative memory. The code works nicely now for the single level. I am going to extend the code to multiple levels in the upcoming weeks so that we will have a very nice framework.

In what follows, I am going to explain the theoretical progress for the journal paper. the MATLAB code for the adaptive neural network as well as the verification of the performance analysis can be found in the accompanying files.

2 Error Correction Performance

For the sake of completeness, the summary of the learning and recall algorithm of the proposed neural associative memory is given in the appendix.

In order to find an analytical bound on the error correction performance, we assume the followings:

- We are given an irregular bipartite graph with right-degree distribution being fixed (i.e. the degree distribution of the pattern nodes are given).
- The graph is assumed to be constructed randomly in the sense that in each round, a pattern node is selected and based on its degree, it connects randomly to constraint nodes.
- We also assume error cancellation does not happen at constraint nodes.
- Finally, we assume the original bit-flipping algorithm is used to correct errors (see appendix or [6]).

Now we take the following steps in order to bound the probability of making an error *in one iteration*:

1. We start by dividing the error in two types: a correct node updates itself mistakenly (P_{e_1}) and a noisy node update itself in the wrong direction (P_{e_2}).
2. We find an explicit relationship for the average of P_{e_1} as a function of the number of nodes in the neighborhood of the noisy pattern nodes.
3. We also find a *bound* on the average of P_{e_2} as a function of the number of nodes in the neighborhood of the noisy pattern nodes.
4. Finally, we find an explicit relationship for the average of the neighborhood size of the nosiy pattern nodes.

2.1 Upper bound on the probability of error

Consider the bit-flipping method in which each pattern neuron decides to update its value or not and if so, in which direction. The decision to update is based on the number of feedbacks a given neuron receives. If the number of feedbacks is larger than a rational number $0 < \gamma < 1$ times the out-degree of the neuron, then the neuron updates itself. Note that the feedbacks could have different signs. The direction of the update is based on the net input value received by the neuron. In other words, if the net input (weighted) sum is positive, the neuron increases its value by one. Otherwise, the value is reduced by 1. If the net input sum is equal to zero, nothing happens.

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 nodes that do not have any connection to any node in $\mathcal{E}_t$. Furthermore, Let $\mathcal{C}_t$ be the set of correct pattern nodes.

Based on the error correcting algorithm and the above notations, at a given iteration two types of error are possible:

1. A node $x \in \mathcal{C}_t$ decides to update its value. Let P_{e_1} denote the probability of this phenomenon.
2. A node $x \in \mathcal{E}_t$ updates its value in the incorrect direction. Let P_{e_2} denote the probability of error for this type.

In what follows, we provide some bounds on the above probabilities of error, with the assumption that *noise cancellation does not occur* in the constraint nodes. [1]

2.1.1 Error probability - type 1

To begin, let P_1^x be the probability that a node $x \in \mathcal{C}_t$ with degree d_x *does not* updates its state (i.e. it makes a correct decision). We have:

$$P_1^x = \Pr\{\frac{|\mathcal{N}(\mathcal{E}_t) \cap \mathcal{N}(x)|}{d_x} \leq \gamma\} \tag{1}$$

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

$$P_x^1 = \sum_{i=0}^{\gamma d_x} \binom{d_x}{i} \left(\frac{S_e}{m}\right)^i \left(1 - \frac{S_e}{m}\right)^{d_x - i} \tag{2}$$

Where $S_e = \mathbb{E}(|\mathcal{N}(\mathcal{E}_t)|)$ is the average neighborhood size of $\mathcal{E}_t$. We will derive an analytic expression for S_e later.

As a result of the above equations, we have:

$$P_{e_1} = 1 - \mathbb{E}_{d_x}(P_1^x) \tag{3}$$

[1] Note that since our noise amplitudes are integer and the weights are pseudo-random real numbers, the probability of noise cancellation, i.e. undetected noise, is extremely small.

2.1.2 Error probability - type 2

A node $x \in \mathcal{E}_t$ makes a wrong decision if the net input sum it receives has a different sign than the sign of noise it experiences. Instead of finding an exact relation, we bound this probability by the probability that the neuron x shares at least half of its neighbors with other neurons, i.e. $P_{e_2} \leq \Pr\{\frac{|\mathcal{N}(\mathcal{E}^*{}_t)\cap\mathcal{N}(x)|}{d_x} \geq 1/2\}$, where $\mathcal{E}^*{}_t = \mathcal{E}^*{}_t \setminus x$. Letting $P_x^2 = 1 - \Pr\{\frac{|\mathcal{N}(\mathcal{E}^*{}_t)\cap\mathcal{N}(x)|}{d_x} \geq 1/2\}$, we will have:

$$P_2^x = \sum_{i=0}^{\lfloor d_x/2 \rfloor} \binom{d_x}{i} \left(\frac{S_e}{m}\right)^i \left(1 - \frac{S_e}{m}\right)^{d_x - i} \tag{4}$$

Where $S_e = \mathbb{E}(|\mathcal{N}(\mathcal{E}_t)|)$ is the average neighborhood size of $\mathcal{E}_t$ and $e = |\mathcal{E}_t|$.

Therefore, we will have:

$$P_{e_2} \leq 1 - \mathbb{E}_{d_x}(P_2^x) \tag{5}$$

2.2 Average neighborhood size

Now it is time to obtain an expression for $S_e = \mathbb{E}(|\mathcal{N}(\mathcal{E}_t)|)$. To do so, 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.

Now let S_e denote the size of the neighborhood of $\mathcal{E}$ when the size of $\mathcal{E}$ is equal to e. We write S_e recursively in terms of e as follows:

$$\begin{aligned} S_{e+1} &= \mathbb{E}_{d_x}\{\sum_{j=0}^{d_x} \binom{d_x}{j} \left(\frac{S_e}{m}\right)^{d_x - j} \left(1 - \frac{S_e}{m}\right)^j (S_e + j)\} \\ &= \mathbb{E}_{d_x}\{S_e + d_x(1 - S_e/m)\} \qquad (6) \\ nonumber & \qquad (7) \\ &= S_e + \bar{d}(1 - S_e/m) \qquad (8) \end{aligned}$$

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. Noting that $S_1 = \bar{d}$, one obtains:

$$S_e = m\left(1 - (1 - \frac{\bar{d}}{m})^t\right) \tag{9}$$

2.2.1 Wrapping up

Combining equations (3) and (3) we will have:

$$\begin{aligned} P_e^x(t) &= \Pr\{x \in \mathcal{C}_t\} P_{e_1} + \Pr\{x \in \mathcal{E}_t\} P_{e_2} \\ &= \frac{n-e}{n} P_{e_1} + \frac{e}{n} P_{e_2} \\ &= 1 - \frac{n-e}{n} \bar{P}_1^x - \frac{e}{n} \bar{P}_2^x \end{aligned} \tag{10}$$

where $\bar{P}_i^x = \mathbb{E}_{d_x}\{P_i^x\}$.

And finally, the average block error rate is given by the probability of at least one pattern node x makes a mistake. Therefore:

$$P_e(t) = 1 - (1 - P_e^x(t))^n \tag{11}$$

2.3 Verification of the analysis

In order to verify the correctness of the above analysis, we have performed some simulation the results of which are given in this section.

Figure 1 illustrates the average neighborhood size in each iteration for a randomly chosen degree distribution with $n = 400$ and $m = 200$ being the number of pattern and constraint nodes, respectively. We generated 100 random graphs and the dashed line shows the average neighborhood size over these graphs. It is obvious that the theoretical value approximates the simulation results rather exactly.

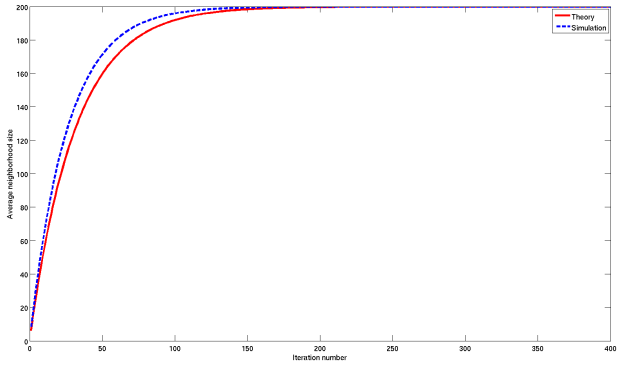

Figure 1: The theoretical estimation and simulation results for the neighborhood size of an irregular graph with a given degree-distribution for $n = 400$, $m = 200$ and over 2000 random graphs.

Figure 2 shows the theoretical bound on the block error rate as well the simulation results. Note that although the upper bound is loose for small number of initial erroneous nodes, it becomes tight as the number of initial noisy nodes grow.Furthermore, even in case of small initial noisy nodes, the bound yields accetable results in terms of block error rate for a neural network. The degree

distribution for this graph was derived from an actual learned neural graph for $n = 100$ and $m = 50$.

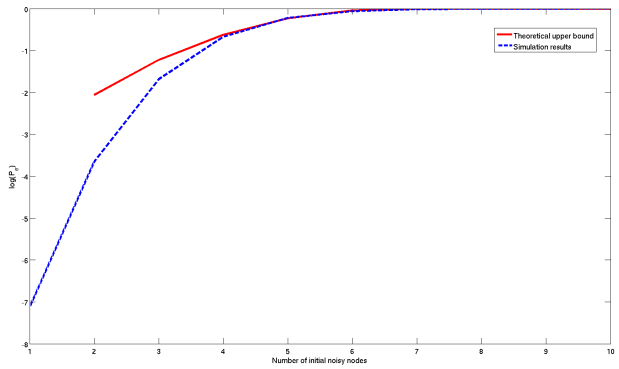

Figure 2: The theoretical bound and simulation results for the block error rate with $n = 100$, $m = 50$. The degree distribution was derived from an actual learned neural graph.

2.4 Some remarks

1. Although the randomness assumption may not be correct due to the fact that the resulting graphs may not be the "parity check" of the given code, but over all ensembles of random codes this assumption might hold. Besides, we really can not analyze the exact error correcting behavior without randomness assumptions unless we are given the neural graph.

2. The above analysis is only the probability of making an error in one iteration as a function of the number of noisy nodes. Simulation results, on the other hand, shows the final probability of error. So part of the reason for not obtaining tight results is due to this discrepancy. We are going to extend the analysis to multiple iterations in the upcoming weeks.

3 Ideas for Proof of Convergence

3.1 Failed attempts

So far, I have worked on the following approaches but with no success yet

- I tried to bound the MSE in each iteration and show that the MSE decreases given that optimization parameters are chosen carefully. But the problem is that the bound will be very loose, resulting in improper simulation parameters. In many other cases, the convergence can not be proved either.

- Then I tried to consider the above argument on MSE, but in the probabilistic form, i.e. show that the MSE decreases with high probability. This was the approach in the previous report which turns out to be trickier than it seems.

3.2 Hopefuly successful path

Despite all the unsuccessful attempts, I have found a book on stochastic approximation techniques [1] which might be helpful. This is actually the book used by [2] and [3] to prove the convergence of their neural algorithm which finds a subspace as well. The idea is that we have formulate the iterative approximation algorithm as an Ordinary Differential Equation (ODE), show that the algorithm converges to the solution of this ODE, find the solution of the ODE and show that it is what we were looking for.

In [2] and [3], the proposed stochastic algorithm can be modeled as a linear ODE, the solution of which is easy to find. In our case, however, due to the sparsity constraint, we will have non-linear terms which complicates things. So we have two possibilities:

1. Find the solution of the non-linear ODE.
2. Using proper conditions on the sparsity constraint, show that the solution of the non-linear version is asymptotically equal to the original linear ODE. Therefore, we can solve the linear ODE and let the algorithm converge to this solution.

At the moment, I am going to follow to second approach as it is much easier than the first one.

Another possibility to prove the convergence of the algorithm is to find an ODE for the MSE itself (instead of the variables). This is similar to the approach use in [4]. However, it is not as easy as [4] given the difference in the nature of the our algorithm with that of [4]. But this approach may turn out useful as well.

4 Future Works

I am going to to do the followings in the upcoming weeks:

1. Work on the proof of the convergence of the learning algorithm, using ideas from [1] and [4].
2. Tighten the bound on the error correction performance of the bit-flipping algorithm and extend to multiple iterations.
3. Complete the code for the adaptive neural associative memory and make it a useful package for further usage.
4. Organize a meeting with the group of Prof. Gerstner to do a sanity check on our adaptive neural model.

A Journal Paper

A.1 Summary of the Learning Algorithm

In order to develop a simple iterative algorithm, we formulate the problem as an optimization framework and then use primal-dual approaches to iteratively find the solution. The problem to find a constraint vector W is given by equation (12).

$$\min \sum_{\mu=1}^{C} |x^{\mu} \cdot w|^2 \tag{12a}$$

Algorithm 1 Iterative Learning

Input: Set of patterns x^μ with $\mu = 1, \ldots, C$, stopping point p.
Output: w
  while $\max_\mu |y(\mu, t)| > p$ **do**
    Compute $y(\mu, t) = x^\mu.w(t)$
    Update $w(t+1) = w(t) - \alpha_t y(\mu, t) \left(x^\mu - \frac{y(\mu,t)w(t)}{\|w(t)\|^2} \right) - \lambda_t f(w(t))$.
    Update $\lambda_{t+1} = \left[\lambda_t + \delta(\epsilon - \|w\|_2^2) \right]$.
    $t \leftarrow t + 1$.
  end while

subject to

$$\|w\|_0 \leq q \tag{12b}$$

and

$$\|w\|_2^2 \geq \epsilon \tag{12c}$$

where $q \in \mathbb{N}$ determines the degree of sparsity and $\epsilon \in \mathbb{R}^+$ prevents the all-zero solution.

Therefore, we first relax (12) as follows:

$$\min \sum_{\mu=1}^{C} |x^\mu \cdot w|^2 + \lambda(g(w) - q'). \tag{13a}$$

subject to:

$$\|w\|_2^2 \geq \epsilon \tag{13b}$$

In the above problem, we have approximated the constraint $\|w\|_0 \leq q$ with $g(w) \leq q'$ since $\|.\|_0$ is not a well-behaved function. The function $g(w)$ is chosen such that it favors sparsity. For instance one can pick $g(w)$ to be $\|.\|_1$, which leads to ℓ_1-norm minimizations. In this paper, we consider the function

$$g(w) = \sum_{i=1}^{n} \tanh(\sigma w_i^2)$$

where σ is chosen appropriately. The larger σ is, the closer $g(w)$ will be to $\|.\|_0$. By calculating the derivative of the objective function and primal-dual optimization techniques we obtain the iterative algorithm given by algorithm 1. Where $f(w) : \mathcal{R}^n \rightarrow \mathcal{R}^n = \nabla g(w)$ is the gradient of the penalty term for non-sparse solutions. Since we are interested in finding m orthogonal vectors, we have to do the above procedure m times in parallel.

References

[1] H. J. Kushner, G. G. Yin, *Stochastic approximation algorithms and applications*, Springer Verlag, 1997.

[2] E. Oja, J. Karhunen, *On stochastic approximation of eigenvectors and eigenvalues of the expectation of a random matrix*, J. Math. Analysis and Applications, Vol. 106, No. 1, 1985, pp 6984.

[3] E. Oja, J. Karhunen, *An analysis of convergence for a learning version of the subspace method*, J. Math. Analysis and Applications, Vol. 91, No. 1, 1983, pp. 102111.

[4] D. L. Donoho, A. Maleki, A. Montanari, *Message passing algorithms for compressed sensing*, Proc. Nat. Acad. Sci., Vol. 106, 2009, pp. 1891418919.

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

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