Progress Report
3 to 17 October, 2010

Amir Hesam Salavati, Raj K. Kumar
E-mail: hesam.salavati@epfl.ch, raj.kumar@epfl.ch

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

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

October 18, 2010

1 Introduction

In our discussion last week, we found out that Hopfield networks can not memorize codewords of a linear code. In other words, if we determine the weights of a Hopfield network according to the classical rule $w_{ij} = \sum_{m=1}^{M} x_i^m x_j^m$, where x_i^m and x_j^m are the i^{th} and j^{th} codebit of a linear code, then $w_{ij} = 0, \forall\ i, j$. To overcome this problem, we considered a modified version of the weighting rule given below:

$$w_{ij} = \frac{1}{M} \sum_{m=1}^{M} \Omega_m x_i^m x_j^m,\ i \neq j \tag{1}$$

Here, Ω_m's are problem parameters that we should determine. The first condition that they should satisfy, the stability condition, is that given the weights according to equation (1) if the network is initialized with one the memorized patterns, it should not evolve. In other words, the memorized patterns must be stable states of the network.

In the next section, we suggest a way to determine Ω_m's such that the codewords of a linear code are stable states of the Hopfield network. If everything works out correctly, then this is equivalent to increasing the storage capacity of Hopfield networks exponentially.

2 Modified Weighting Rule

In a Hopfield networks, neurons update their status according to equation (2).

$$x_i = \begin{cases} 1, & \sum_{j=1}^{n} w_{ij} x_j > \theta \\ -1, & \text{Otherwise} \end{cases} \tag{2}$$

Where x_i is the status of the i^{th} neuron, θ is the firing threshold and n is the number of neurons (nodes). Now suppose the weights are determined according to equation (1). The stability condition means that if we initialize the neurons states with a memorized pattern, say $\underline{X}^{s1}$, then:

$$x_i^s = 1 \iff \sum_{j=1,\neq i}^{n} w_{ij} x_j^s > \theta \tag{3}$$

[1] Underline denotes vector.

Expanding the above equation we obtain:

$$x_i^s = 1 \iff \frac{1}{M} \sum_{j=1,\neq i}^{n} x_j^s \sum_{m=1}^{M} \Omega_m x_i^m x_j^m > \theta$$

Noting that x_i^s is the i^{th} codebit of the s^{th} message pattern, we can rewrite the above equation as

$$(-1)^{\lambda_i(\underline{Y}^s)} = 1 \iff \frac{1}{M} \sum_{j=1,\neq i}^{n} (-1)^{\lambda_j(\underline{Y}^s)} \sum_{m=1}^{M} \Omega_m (-1)^{\lambda_i(Y^m)} (-1)^{\lambda_j(Y^m)} > \theta \quad (4)$$

where $\underline{Y}^s$ is the s^{th} k-bit message pattern and $\lambda_i(Y)$ is a linear combination of k message bits that determines the value of the i^{th} code bit. Now let's expand the right-hand sum in equation (4). We get:

$$\begin{aligned}
\mathcal{S} &= \frac{1}{M} \sum_{j=1,\neq i}^{n} (-1)^{\lambda_j(\underline{Y}^s)} \sum_{m=1}^{M} \Omega_m (-1)^{\lambda_i(Y^m)} (-1)^{\lambda_j(Y^m)} \\
&= \frac{1}{M} \sum_{j=1,\neq i}^{n} (-1)^{\lambda_j(\underline{Y}^s)} \Omega_s (-1)^{\lambda_i(\underline{Y}^s)} (-1)^{\lambda_j(\underline{Y}^s)} \\
&+ \frac{1}{M} \sum_{j=1,\neq i}^{n} (-1)^{\lambda_j(\underline{Y}^s)} \sum_{m=1,\neq s}^{M} \Omega_m (-1)^{\lambda_i(Y^m)} (-1)^{\lambda_j(Y^m)} \\
&= \frac{(n-1)\Omega_s}{M} (-1)^{\lambda_i(\underline{Y}^s)} + \frac{1}{M} \sum_{j=1,\neq i}^{n} (-1)^{\lambda_j(\underline{Y}^s)} \sum_{m=1,\neq s}^{M} \Omega_m (-1)^{(\lambda_i+\lambda_j)(Y^m)} \quad (5)
\end{aligned}$$

As a result, in order to satisfy condition (4), we must have:

$$(-1)^{\lambda_i(\underline{Y}^s)} = 1 \iff \mathcal{S}' = \sum_{j=1,\neq i}^{n} (-1)^{\lambda_j(\underline{Y}^s)} \sum_{m=1,\neq s}^{M} \Omega_m (-1)^{(\lambda_i+\lambda_j)(Y^m)} > \theta' \quad (6)$$

Where $\theta' = M\theta - (n-1)\Omega_s(-1)^{\lambda_i(\underline{Y}^s)}$.

Hence, the goal is to determine Ω_m's such that equation (6) is satisfied for all input patterns. One can run an optimization and find the appropriate Ω_m's. However, if we follow a probabilistic approach, we can find the probability that $(\lambda_i + \lambda_j)(Y^m) = 0$, for two code bits i and j. Then, we might find a simpler criteria to calculate Ω_m.

2.1 Calculating Probability of Having Two Similar Code Bits

To calculate the probability of having $(\lambda_i+\lambda_j)(Y^m)=0$, suppose we have an LDGM code with node degree distribution (L,R), i.e. L_i (γ_i) is the fraction of message (code) nodes with degree i. Furthermore, assume Y^m has weight ω_m. Without loss of generality, we assume that the first ω_m elements of Y^m are equal to 1 and the rest are 0. As a result, $(\lambda_i+\lambda_j)(Y^m)=0$ if and only the linear combination if$(\lambda_i+\lambda_j)(Y)$ has an even number of 1's in the first ω_m places. In other words:

$$P_{even}^{\omega_m}=\Pr\{(\lambda_i+\lambda_j)(Y^m)=0;|Y^m|=\omega_m\}=\sum_{\ell=0}^{\lfloor\omega_m/2\rfloor}\binom{\omega_m}{2\ell}p_1^{2\ell}(1-p_1)^{\omega_m-2\ell}=\frac{1+(1-2p_1)^{\omega_m}}{2} \tag{7}$$

Where p_1 is the probability that the linear combination $(\lambda_i+\lambda_j)(Y)$ is equal to 1 in a given position, i.e. if $(\lambda_i+\lambda_j)(Y)=y_1+y_3$ then the linear combination has two non-zero coefficients at the first and third places. To derive p_1, let's assume that the code nodes i and j has degrees γ_j and γ_j. As a result, $(\lambda_i+\lambda_j)(Y)$ has a coefficient equal to 1 if either one of λ_i or λ_j has a nonzero coefficient at that place and the other one has a zero there. Hence:

$$p_1(\gamma_i,\gamma_j)=\frac{\gamma_i}{k}(1-\frac{\gamma_j}{k})+(1-\frac{\gamma_i}{k})\frac{\gamma_j}{k}$$

Averaging over right degree distributions we get:

$$p_1=2\frac{\bar{\Gamma}}{k}(1-\frac{\bar{\Gamma}}{k}) \tag{8}$$

In which $\bar{\Gamma}$ is the average right-degree. Combining equations (9) and (8) we obtain the following equation for $P_{even}(\omega_m)$:

$$P_{even}(\omega_m)=\frac{1+(1-2\bar{\Gamma}/k)^{2\omega_m}}{2} \tag{9}$$

2.2 Deriving Ω_m

Now that we have $P_{even}(\omega_m)$, we make a simplifying assumption: all patterns with the same message weight have equal Ω_m's. In other words:

$$\Omega_\omega=\Omega_m=\Omega_{m'}\iff\omega_m=\omega_{m'}=\omega$$

Furthermore, since we know that the number of patterns with weight ω is simply $\binom{k}{\omega}$, and we have $P_{even}(\omega)$, we can assume that on average, $\binom{k}{\omega}P_{even}(\omega)$ of all patterns with weight ω has $(-1)^(\lambda_i + \lambda_j)(Y^m) = 1$ and the rest have $(-1)^(\lambda_i + \lambda_j)(Y^m) = -1$. As a result, one can rewrite equation (6) as:

$$(-1)^{\lambda_i(\underline{Y}^s)} = 1 \iff \mathcal{S}' = \sum_{j=1,\neq i}^{n} (-1)^{\lambda_j(\underline{Y}^s)} \sum_{\omega=0}^{k} \Omega_\omega \left(2P_{even}(\omega) - 1\right) \left(\binom{k}{\omega} - \delta(\omega - \omega_s)\right) > \theta'$$

where $\omega_s = |\underline{Y}^s|$ and $\delta(.)$ is the Diract delta function. The above relationship simplifies to:

$$(-1)^{\lambda_i(\underline{Y}^s)} = 1 \iff \mathcal{S}" = \left(1 - 2\bar{\Gamma}/k\right)^{2\omega_s} \left[\sum_{\omega=0}^{k} \Omega_\omega \left(1 - 2\bar{\Gamma}/k\right)^{2\omega} \binom{k}{\omega} - \Omega_{\omega_s} \left(1 - 2\bar{\Gamma}/k\right)^{2\omega_s}\right] \quad (10)$$

$$> \frac{M\theta}{n-1} - (-1)^{\lambda_i(\underline{Y}^s)} \Omega_{\omega_s} \quad (11)$$

Repeating the above process for $k+1$ patterns with weights from 0 to k, we get a matrix form of equation (10) as follows:

$$A\underline{\Omega} > \frac{M\theta}{n-1}\underline{1} - B\underline{\Omega} \quad (12)$$

Where $\underline{\Omega} = \{\Omega_0, \ldots, \Omega_k\}$ and $\underline{1}$ is the all one vector. Furthermore, matrices A and B are given below:

$$A = \begin{pmatrix} (1-2\bar{\Gamma}/k)^0(\binom{k}{0}-1) & (1-2\bar{\Gamma}/k)^2\binom{k}{1} & \ldots & (1-2\bar{\Gamma}/k)^{2k}\binom{k}{k} \\ (1-2\bar{\Gamma}/k)^2\binom{k}{0} & (1-2\bar{\Gamma}/k)^4(\binom{k}{1}-1) & \ldots & (1-2\bar{\Gamma}/k)^{2k+2}\binom{k}{k} \\ \vdots & \vdots & \vdots & \vdots \\ (1-2\bar{\Gamma}/k)^{2k}\binom{k}{0} & (1-2\bar{\Gamma}/k)^{2+2k}\binom{k}{1} & \ldots & ((1-2\bar{\Gamma}/k)^{2k+2k}\binom{k}{k}-1) \end{pmatrix}$$

and

$$B = \begin{pmatrix} (-1)^{\lambda_i(Y^0)} & 0 & \ldots & 0 \\ 0 & (-1)^{\lambda_i(Y^1)} & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & (-1)^{\lambda_i(Y^k)} \end{pmatrix}$$

If we assume $\theta = 0$, then one notes that in order to satisfy the stability condition, we just need to have find $\underline{\Omega}$ such that:

$$-\underline{\Omega} < A\underline{\Omega} < \underline{\Omega},\ \underline{\Omega} > \underline{0} \quad (13)$$

Because in this case, if $(-1)^{\lambda_i(\underline{Y}^s)} = 1$ for some pattern s, then equation (10) is satisfied if unequality (13) holds. Similar argument is valid for the case where $(-1)^{\lambda_i(\underline{Y}^s)} = -1$.

An easy way to satisfy equation (13) is to pick $\underline{\Omega}$ such that it is an all-positive eigenvector of A with a positive eigenvalue smaller than 1. However, A might be such that it does not have an all positive eigenvector and some elements are equal to zero. Therefore, in geneal one must use linear programming to find the suitable weights.

3 Future Works

In this report, we have proposed an approach to determine Ω_m for modified Hopfield weighting rule. The suggested method is based on analyzing the stability condition probabilistically. The goal is to show that having chosen Ω_m according to the suggested method, we can memorize 2^k codewords of a linear (n, k) code. Our next step is to use numerical approaches to see if the proposed weighting rule is capable of memorizing 2^k patterns for a reasonably large k.