Learning Linear Codes with Hopfield networks

Hesam & Raj

In this report, we consider the case where a Hopfield network learns and recalls codewords of a linear code (as opposed to learning purely random patterns, which is the case considered in the traditional Hopfield model). By imposing structure on the patterns that are memorized, we expect that the capacity of the Hopfield network will increase significantly, owing to better distance properties between the memorized patterns. The biological relevance of this framework needs to be further investigated: one possible explanation could be a part of the brain that does this mapping from purely random patterns to codewords, before storing them in the brain. One would need to see if implementing the circuitry of an encoder using neurons is reasonable in terms of complexity.

In the sequel, we follow the notation of [1]. The weights in the n-neuron Hopfield network are computed according to

$$w_{ij} = \frac{1}{n} \sum_{m=1}^{2^k} \xi_i^m \xi_j^m, \tag{1}$$

where ξ_i^m denotes the i^{th} symbol of pattern m. Recall that each $\xi_i^m \in \{+1, -1\}$ in the Hopfield network setting.

Define the $2^k \times n$ matrix Ξ of patterns such that the entries $\Xi_{m,j} = \xi_j^m$. We may rewrite (1) by defining the matrix of weights $W = [w_i j]$ as

$$W = \frac{1}{n} \Xi^{\mathsf{T}} \Xi. \tag{2}$$

Let G denote the $k \times n$ generator matrix of a linear code, whose codewords the Hopfield network wishes to learn. Define A to be the $2^k \times k$ integer matrix whose rows comprise of all binary k-tuples

$$A \triangleq \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & & & & \vdots \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}.$$

Define the codeword matrix

$$\Xi_2 \triangleq (AG) \mod 2.$$

Notice that we are considering both A and G to be real matrices and performing integer arithmetic in the above matrix product, followed by reduction mod 2 to obtain Ξ_2 (for reasons that will become clear in the sequel). We now define the matrix of patterns Ξ corresponding to the Hopfield network to be

$$\Xi = 2\Xi_2 - \mathfrak{I},$$

where $\mathfrak{I}$ is the all ones matrix, and compute the weights for the Hopfield network as in (2). We hence obtain

$$\begin{aligned} W &= \frac{1}{n}(2\Xi_2 - \mathfrak{I})^{\mathsf{T}}(2\Xi_2 - \mathfrak{I}) \\ &= \frac{1}{n}\left[4\Xi_2^{\mathsf{T}}\Xi_2 - 2\Xi_2^{\mathsf{T}}\mathfrak{I} - 2\mathfrak{I}^{\mathsf{T}}\Xi_2 + \mathfrak{I}^{\mathsf{T}}\mathfrak{I}\right] \end{aligned}$$

Notice that $\mathfrak{I}^{\mathsf{T}}\mathfrak{I} = 2^k\mathfrak{I}_{n\times n}$, and that $\Xi_2^{\mathsf{T}}\mathfrak{I} = \mathfrak{I}^{\mathsf{T}}\Xi = 2^{k-1}\mathfrak{I}_{n\times n}$. We hence obtain that

$$W = \frac{1}{n}\left[4\Xi_2^{\mathsf{T}}\Xi_2 - 2^k\mathfrak{I}_{n\times n}\right].$$

We now need to relate $\Xi_2^{\mathsf{T}}\Xi_2$ in terms of the parameters of the linear code that we choose, say for example we choose a d-regular LDGM ensemble. Computing $\Xi_2^{\mathsf{T}}\Xi_2$ reduces to answering the following question: In how many places do two codewords from our linear code both have ones? This is something that we need to work on.

REFERENCES

[1] J. Hertz, A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Westview Press, 1991.