ERC Report - July 8, 2010

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)

July 8, 2010

1 Introduction

Based on the discussion in my candidacy exam, I have started working on modeling dynamical behavior of associative memory which is the mechanism in brain corresponding to error correction and pattern matching. In order to analyze associative memory, it is more convenient to start with Hopfiled networks and then extend the analysis to real neuronal networks. Hopfield networks are artificial neural networks inspired from the ability of neuronal networks to memorize and recall patterns.

In what follows, I first review principles of Hopfield networks. A framework is then suggested to extend coding theoretical arguments used in investigation of belief propagation algorithm to analyze associative memory.

2 Hopfield Networks

Hopfiled networks are artificial neural networks used to store and memorize binary patterns [1]. A Hopfield network is composed of n neurons. Each neuron is characterized by a state function, S_i for the i^{th} neuron, which determines if a neuron is firing, i.e. $S_i = 1$, or not, $S_i = 0$.

In a Hopfield network, neurons are connected to each other and can affect the firing of their neighbors. This effect is modeled by directed edges in the neural network. An edge from neuron i to neuron j indicates that the output of neuron i affects the state of neuron j. Each connection has also a weight, w_{ij}, which represents the strength of the connection.

Similar to other neural networks, In a Hopfield network a neuron fires if the weighed sum of its inputs crosses a threshold, θ_i. Therefore, the state of neuron i, N_i, is given by the following equation:

$$S_i = U(\sum_{j \in \Pi_i} w_{ji} S_j - \theta_i) \tag{1}$$

where $U(.)$ is the step function and Π_i is the set of neurons having a directed edge toward N_i. Given an initial state and a set of weights, the network evolves by sequentially updating neurons states according to the above equation.

Initially, all n neurons are connected to each other and all weights are initialized to an arbitrary amount. In the original Hopfield model, the goal is to modify the weights such that the network is able to memorize p randomly chosen patterns such that whenever one of the patterns is given to the network as the initial state, after a few iterations the corresponding output pattern is given. It is shown that if weights are chosen according to the following rule, the network is guaranteed to correctly recall the p memorized patterns if p/n is less than the network capacity (which is around 0.14 [1]).

$$W = \frac{1}{n} \sum_{k=1}^{p} \underline{S}^T \underline{S} \tag{2}$$

In the above equation, W is the matrix of weights whose (i, j) element is equal to w_{ij} and $\underline{S}$ is the row vector of all neurons states, i.e. $\underline{S} = \{S_1, \ldots, S_n\}$.

While Hopfield networks are useful in memorizing random patterns, and hence helpful in analyzing similar aspects of associative memory, the model does not address the issue of noisy input patterns directly. Moreover, since it is assumed that patterns are chosen randomly, the capacity of the network, i.e. p/n, is very small (around 0.14). By carefully selecting patterns, such that their minimum distance is not small, one might be able to achieve larger capacities. Moreover, investigating the performance of Hopfield networks from a coding theoretic point of view is another interesting topic which is more explored in the next section.

3 Hopfield Networks and Belief Propagation

The network used in Hopfield models could be easily transformed to a bipartite graph capturing the effect of different neurons on each other (this is exactly the same method as the one used to transform genetic graphs to gene factor graphs in gene regulatory networks). The dynamical behavior of the Hopfield network is very similar to the belief propagation algorithm: in each iteration, a given neuron evaluates the weighted sum of its inputs and compare to a threshold. If the threshold was crossed, neuron fires and remain silent otherwise. Similarly, in a variant of the belief propagation algorithm, a variable node changes its state only if the majority of its neighbors return the same value in that iteration. The majority is determined by a threshold, i.e. if a number of neighboring check nodes greater than a threshold all agree on the same value, the variable node changes its state to the accepted value. In the neural network terms, this is equivalent to calculating the weighed sum and comparing with the threshold when weights are binary.

Therefore, the main difference lies in the fact that Hopfield networks are weighted while the corresponding graph in iterative codes are weightless. Extending belief propagation algorithm to weighted graph seems interesting in analyzing dynamical behavior of Hopfield networks and associative memory. A simple way to capture the weights is to consider them in update rules. For instance, a variable node changes its status when the weighted sum of its neighbors' values crosses a threshold.

4 Conclusions and Future Works

Based on the discussion above, the BP algorithm and the dynamical behavior of the Hopfiled networks share a lot of similarities. The main difference though is that Hopfiled networks are weighted graphs. Therefore, analyzing the performance of the BP algorithm over weighted graphs seems a natural step in analyzing associative memory from coding theoretical point of view. Further, calculating different performance measures, such as probability of errors and convergence speed, is another interesting subjects in this regard which requires extending the methods for analyzing BP algorithm, such as density evolution, to weighted graphs.

The above problems only consider the case when the graph is given and the weights are fixed. However, having a weighted graph could give us more

degree of freedom in code design as well and this corresponds to determining the weights such the performance is optimized. This is essentially the same process performed in brain: weights vary constantly such that the optimal decoder for the given patterns are obtained. The same procedure may help us in designing better codes.

References

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