Hopfield Net

Conner Brinkley

03.24.2020

Biologically-Inspired Computing

The University of Tennessee

INTRODUCTION

The purpose here is to investigate the associative memory capacity of a Hopfield network. This was done by creating a simulator that computes the local fields of 50 randomly assigned patterns, determines the next state of the patterns based on this field, and then compares them to the original neural network to determine if it is stable or not. The resulting raw data is an average over 100 separate runs and was outputted as CSV files to make it easier to graph in Excel.

PROCESS (& THEORY)

The simulator starts with a 50 x 100 matrix that represents 50 different patterns, each consisting of 100 neurons. Every neuron for each run was randomly initialized to either a +1 or -1 state. The first step to calculating the next state of the patterns is to determine the weight associated with each neuron *i* and *j* (w_{ij}), which can be found with the following equation:

$$w_{ij} = \begin{cases} \frac{1}{N}\sum_{k=1}^{p} s_i s_j & i \neq j \\ 0 & i = j \end{cases}$$

Figure 1. No self-coupling equation to calculate the weight associated with neurons *i* and *j*.

All of the weights are stored in a 100 x 100 matrix, where *N* is the number of neurons (in this case it is 100), *p* is the number of patterns (50), and s_i and s_j are the state values of neurons *i* and *j*, respectively (+1 or -1). These weight values are used to determine each neuron's local field, given by the equation below.

$$h_i = \sum_{j=1}^{N} w_{ij} s_j$$

Figure 2. Local field formula for neuron *i*.

The local field values are temporarily stored in a 100-element array, where w_{ij} is the corresponding weight value for neurons *i* and *j* and s_j is the current neural network's state value for neuron *j* (either +1 or -1). The current neural network is determined by the current pattern that is being tested for stability, and the next state of the pattern is determined by using the previously calculated local field (h_i) in the following equations.

$$s'_i = \sigma(h_i)$$

Figure 3. Determines the next state of the neuron *i.*

The next state for each neuron is given by the following conditional equation:

$$\sigma(h_i) = \begin{cases} -1, & h_i < 0 \\ +1, & h_i \geq 0 \end{cases}$$

Figure 4. Uses the local field to determine the next state of neuron *i.*

Put simply, if neuron *i*'s local field value is less than zero, then its new state is -1, and if its local field value is greater than or equal to zero, then its new state is +1.

If any of the new state values differ from the original pattern's neurons, then the entire pattern assigned to the neural network is not stable. Otherwise, it is stable. For each number of patterns imprinted, a counter is kept keeping track of the number of stable ones.

Finally, to compute the probability of stable imprints for each number of patterns, the simulator divides the number of stable ones by the total number of patterns for that iteration. To get the inverse, or the probability of unstable imprints for each number of patterns, the program just subtracts the previous value from one.

SIMULATOR

The program was written in C++ with the help of Dr. Van Hornweder's write up, and the source code can be found in the hopfieldNet.cpp file in the source directory. Due to the amount of runs the data was averaged over, C++ seemed to make the most sense because of its speed. The simulator created two CSV data files, one for each graph: the number of stable imprints vs. the number of imprints and the probability of unstable imprints vs. the number of imprints.

It was not necessary to evaluate different neuron and pattern values for this project, so all of the values were hard coded into the Hopfield network object. However, it was written such that it could be easily modified in the future to accept different values by user input, which could be interesting to investigate at a later date. For the time being, the program is simply run by the following command after compiling with the given makefile:

USAGE: ./hopfieldNet

Once the program finished calculating everything, it outputted all of the data as CSV files, which were then saved as Excel workbooks to graph the results using Excel's plotting tool.

RESULTS

The purpose of all of this was to investigate the associative memory capacity of a Hopfield network. The two graphs that were produced can be used for this analysis. Below is the first graph, which shows the number of stable imprints (y-axis) that result after imprinting *p* patterns (x-axis).

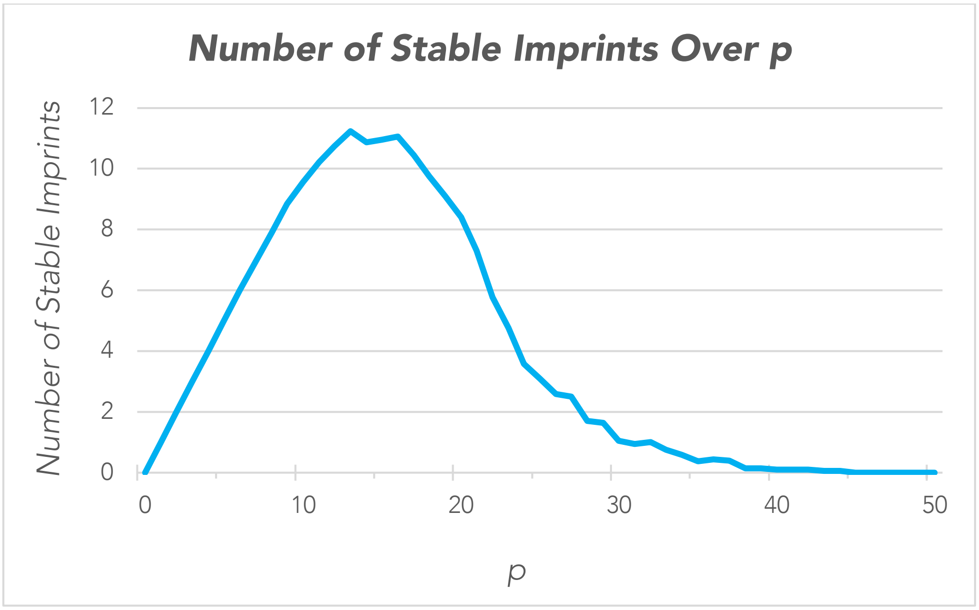

Figure 5.   Graph of stable imprints calculated by the simulator.

The graph above shows a relatively linear relationship, up until around 9 patterns, before quickly peaking at 13 and declining until reaching a minimum around 45 patterns. In simple terms, this basically means that up to 9, every pattern is stable. Even though the number of stable patterns peaks at 13, there is still a greater chance that any one pattern may become unstable than if only

9 patterns were considered. This is supported in the second graph, which shows the percent of unstable imprints (y-axis) that result after imprinting *p* patterns (x-axis).

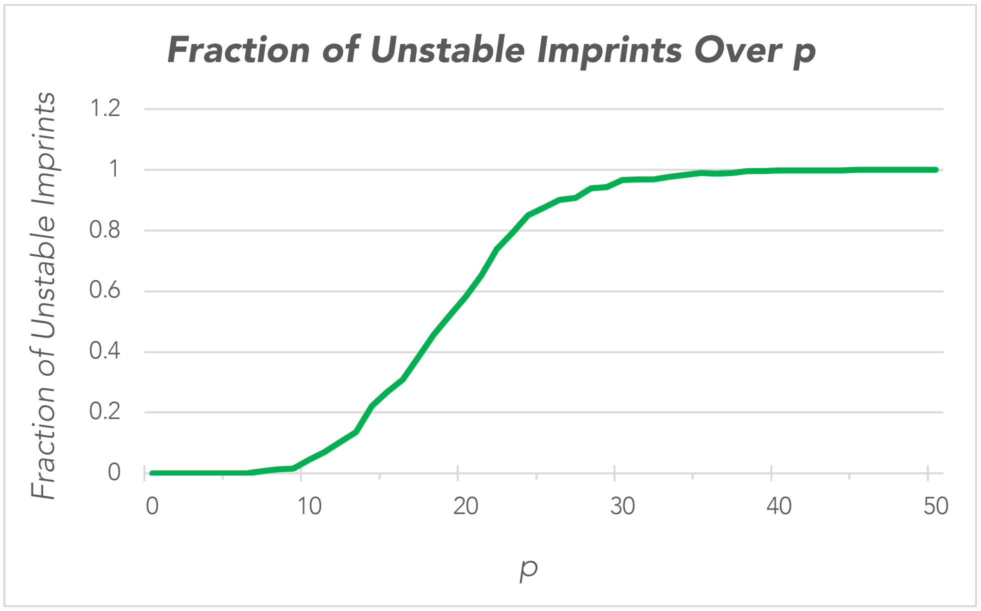

Figure 6. Graph of the probability of unstable imprints calculated by the simulator.

This graph backs up the previous statement. As *p* increases, the percent of unstable imprints also increases. The percent chance of getting an unstable imprint is virtually zero up until the number of patterns reaches 9, at which point it rises exponentially.

CONCLUSION

In terms of associative memory capacity, the lower the chance of an unstable imprint, the better for storage. When *p* is 13, it seems like the capacity for memory is at its peak. However, this capacity is a trade-off for reliability because after 9 patterns, the fraction of unstable imprints grows exponentially. While the fraction of unstable imprints for 13 is still relatively low, this may not be a good thing when trying to preserve exact memory. For reliability and lossless memory, 9 seems to be the sweet spot.

In the future it might be interesting to modify the simulator to take different *p* and *N* values to see how this affects the fraction of unstable imprints over the number of patterns and analyze the results with respect to associative memory capacity.