Project 3: Hopfield Networks

Samuel Steinberg

CS420

March 7th, 2019

Introduction:

This project is the simulation of a recurrent neural network in the form of a Hopfield Network, where the goal is to investigate associative memory capacity. In Hopfield Networks there is no self-coupling, weights are symmetric, and the bipolar states are determined by discontinuous bipolar activation. Convergence is guaranteed, and the aforementioned weights describe strength of connectivity (no self-action allowed). In this project cells are updated asynchronously, and weights use the Hebbian rule for calculations. The program for running the network was written in Java.

Theory and Methods:

The program works by accepting a number of simulations to perform. In this experiment 15 were made. 50 bipolar vectors are then generated, each 100 elements in length (each element is a 1 or -1 and represents an artificial neuron). For each bipolar vector, an array of 100 elements representing the neural network is set to the vector pattern array at the current index. The update state of the current neuron is then determined by summing the weight matrix and the neural network. If the update state is greater than or equal to 0, the next state value is 1. If less than 0, the next state will be -1. This transition from update to next state represents the new state. If the neural net differs from the new state value, the imprinted pattern is unstable. If they are equivalent, they are stable. The number of stable imprinted patterns for each bipolar vector and the fraction of unstable patterns are kept track of using arrays of 50 elements (one for each bipolar vector). After these averages are computed they are written to a spreadsheet where the graphs are generated.

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

Figure 1: Weight Calculation

The weight calculation formula in Figure 1 calculates the strength of connectivity between two neurons *i* and *j*. The weight matrix is symmetric: neuron *i* affects neuron *j* as much as neuron *j* affects neuron *i* ($w_{ij} = w_{ji}$). The sum of the pattern at *i* multiplied by the pattern at *j* is taken through each pattern 1 to 50 (index k). This is divided by the number of neurons (N = 100).

$$W = \begin{pmatrix} 0 & w_{12} & \dots & w_{1i} & \dots & w_{1n} \\ w_{21} & 0 & \dots & w_{2i} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ w_{i1} & w_{i2} & \dots & 0 & \dots & w_{in} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \dots & w_{ni} & \dots & 0 \end{pmatrix}$$

Figure 2: Representation of the weight matrix. Note the diagonals are zeroed, this is because there is no self-coupling.

Since there is no self-coupling, when *i* and *j* are equal the weight is set to 0. This is because no neuron can connect to itself. This is represented visually in Figure 2.

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

Figure 3: Local Field Calculation

The local field is computed by the summation of the weight matrix at a neuron *i* and *j* and the bipolar vector pattern at index *j*. The index *j* represents the iteration through each value of the neural net (this is set to the bipolar vector pattern). The multiplication of the neurons state value and the weight matrix will yield h_i, which represents the update value into the next state of the current neuron. See Figure 4 above for a visual representation.

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

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

Figure 4: Next State Calculation

The next state is determined depending on the result of the update value from the local field computation in Figure 3. If the update value is less than 0, the next state will be -1. If it is greater than or equal to 0 it will be 1. The next state value will be used in comparison to the neurons current state. If they are the same, the imprinted pattern is stable. If they differ, it is an unstable pattern.

Results and Discussion:

After the collection of data, two graphs were generated. One plotted the number of stable imprints over *p* (patterns 1-50), and another plotted the fraction of unstable imprints over *p*. There were 15 simulations run, and the graphs represent the average data over the runs.

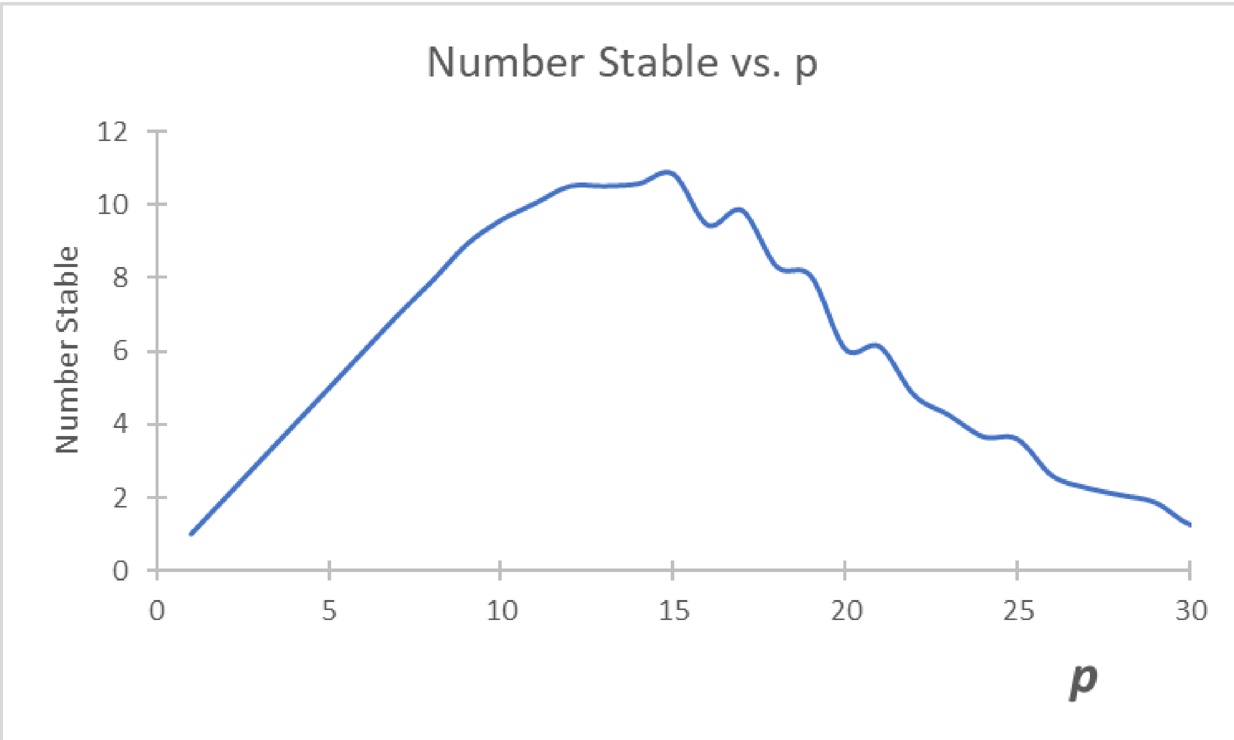

Figure 5: Note the linear increase over the first ~10 iterations, then stable behavior before becoming chaotic at around iteration 15.

Figure 5 represents the number of stable iterations over *p*. Since this experiment investigates associative memory capacity, it makes sense that there is a near linear increase in stable patterns for about the first 10 imprints. This makes sense, since there are not many patterns to remember. The first 10 imprints are quite stable throughout each simulation and it may be concluded the network is reliable up to this point. After this, a high point in the number of stable iterations was reached around imprint 15, after which the numbers become chaotic and decreasing. It can be said that reliability is much less reliable after 15 imprints. This would be expected, as it is much easier for a person to recall a few pictures or other visuals, for example, rather than a few dozen.

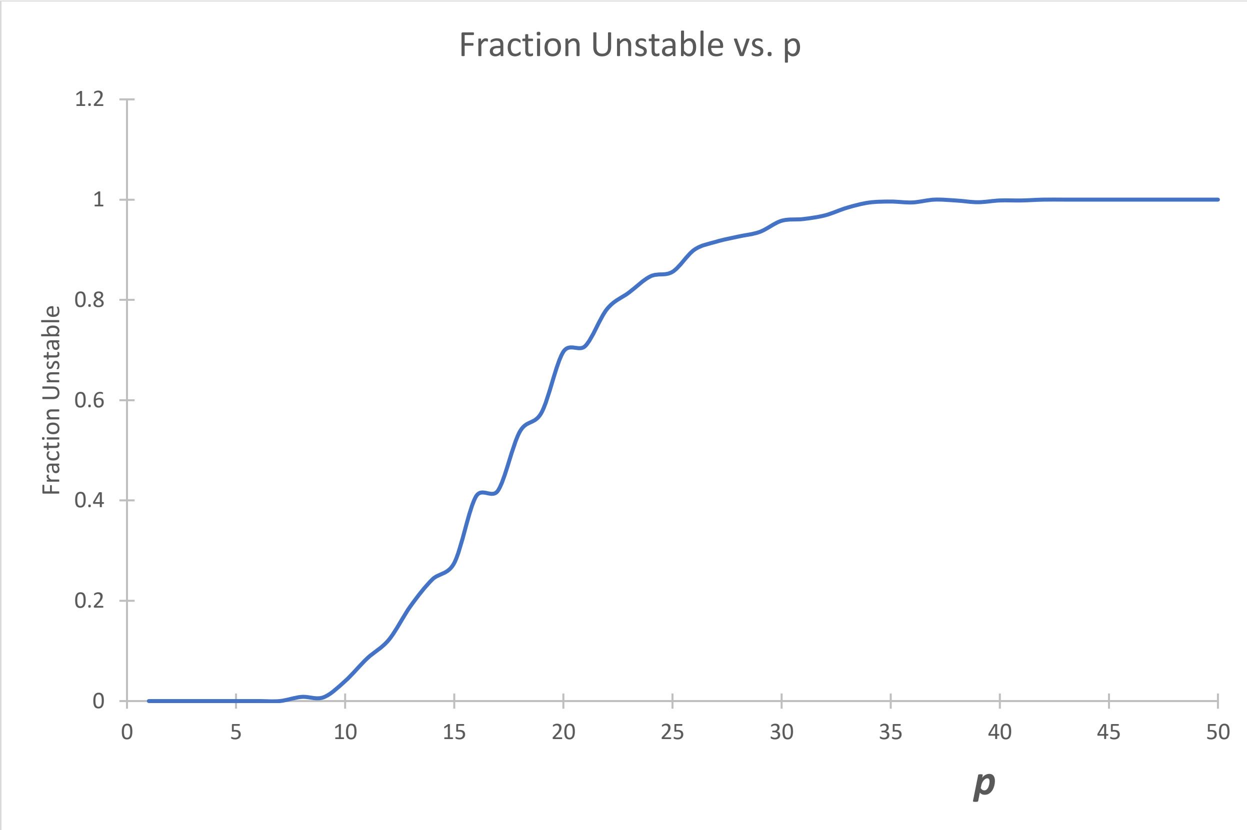

Figure 6: Note that the fraction of unstable imprints is related inversely to Figure 5 and the number of stable imprints.

Figure 6 represents the fraction of unstable imprints over *p*. This is the probability that a pattern is unstable. This further establishes the claims made when examining Figure 5, where the stability is relatively reliable over the first 10 imprints. Over each simulation there were almost 0 unstable imprints over this period between them. Reliability also began to decrease after this period and became increasingly erratic over the following simulations where it became almost impossible to recover imprinted patterns at around imprint 33. Similarly, the associative memory capacity is basically non-existent by this point (recovery extremely unlikely/impossible).

Conclusions:

This project investigated the associative memory capacity of a recurrent neural network: a Hopfield Network. In addition to calculating valuable parameters including connectivity weights, local fields, next state calculations, stable imprints, and fraction of unstable imprints, the program also generates two graphs. One is for the number of stable imprints over *p*, and the other represents the fraction of unstable imprints over *p*. These visual representations in Figure 5 and 6 show that pattern imprints were reliably stable over the first 10 imprints, reached a high point around imprint 15, and then became erratic and declining until there were no stable imprints around iteration 35. This would be expected with associative memory capacity, as it is easier to recall a couple memories than a few dozen. The capacity reached a point around imprint 30 where memory recovery became extremely unlikely and eventually was consistently unstable.