Christopher Moran

Dr. MacLennon

CS 420

24 March 2020

Project 3: Hopfield Net

Introduction

The purpose of this assignment was to investigate the associative memory capacity of a Hopfield network, specifically with 100 neurons without self-coupling. No external data was used in this assignment, as everything was able to be generated from within the program itself. That being said, the program does generate a csv that holds the data determining the probabilities of each net.

Methodology

The first thing that had to be done here was to generate a 2d array, with the master array being 50 units long, with each unit commanding a vector of 100 units in length, and where each of these units is set randomly to either a 1 or -1. These vectors compose the neural nets. As the program moves through the array, it uses an index named P to do so as p is set to increasing amounts, starting at 0 and ending at 49, coming to a total of 50 iterations. There are a number of things that need to be done for each iteration, first of which is where the patterns are imprinted. This means that there needs to be a pattern imprinted for each neural net. To do this, a weights matrix must be created. This matrix is a 100x100 2d array that is assigned to each 100-length vector/pattern. The values for these weights are determined by this equation:

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

It should be noted that in this equation, W refers to the weight matrix, N to the total number of neurons, 100, and p to whatever index p is currently being calculated. K is just an index going up to p, and S simply refers to the pattern as extracted by the 50x100 2d array created earlier. This should give the weights matrix.

Once that is completed, the imprinted patterns must be tested. To do this, the initial state/ pattern of whatever index p is currently being calculated must have its new state calculated. This can be done via this equation:

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

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

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

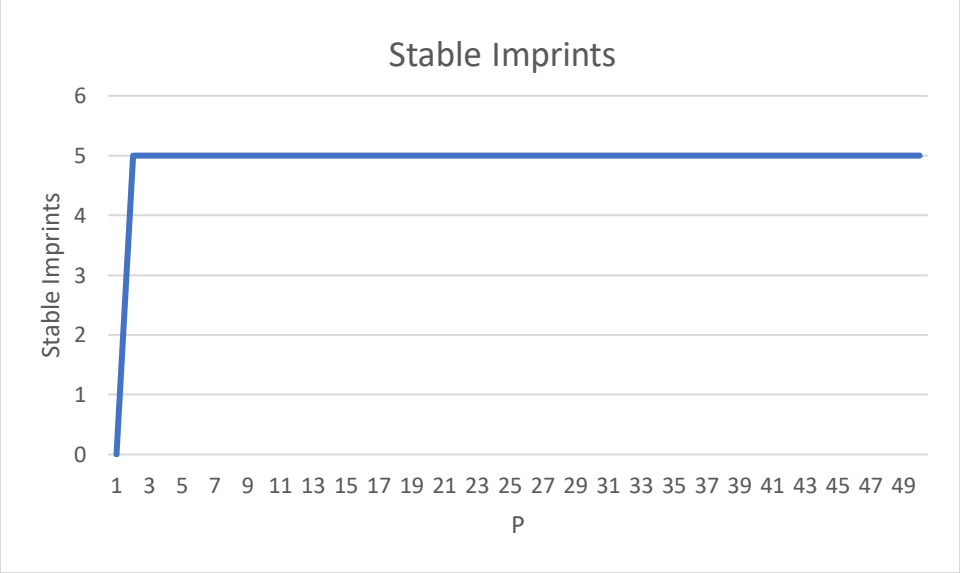
In the first equation here, h refers to the new state that is to be created. J is just an index that goes up to N, set as 100. W refers to its respective weights matrix, and s refers to each of those 1s or -1s, as found in the initial pattern/neural net, and specified by its subscripted index. The next two equations basically say that for each element within the new state array that is greater than or equal to 0, it must now be set to 1. All other units must be set to -1.

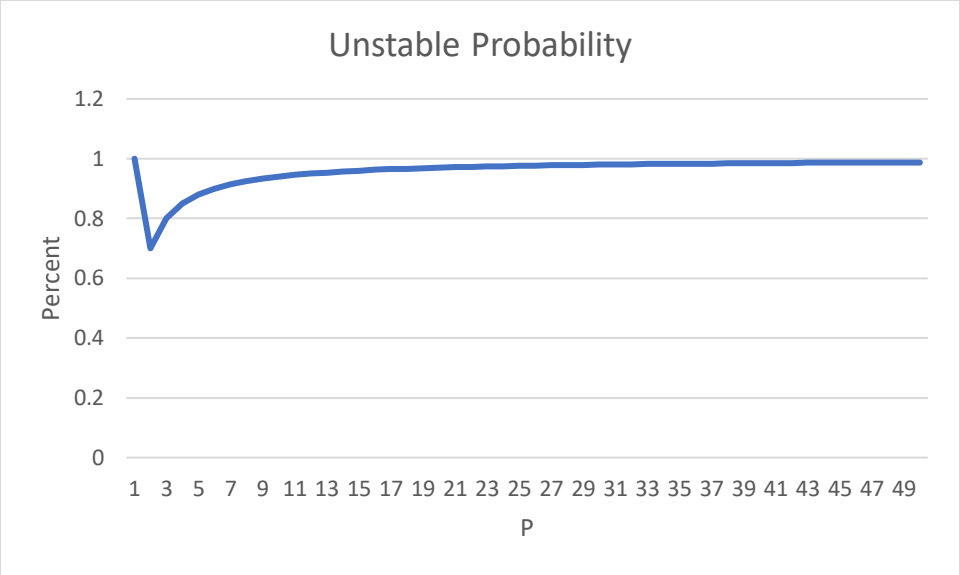
Once a new state array has been created, it must be tested against the neural net it was created from. If they match exactly, then the imprint is said to be stable. If not, then the imprint is unstable.

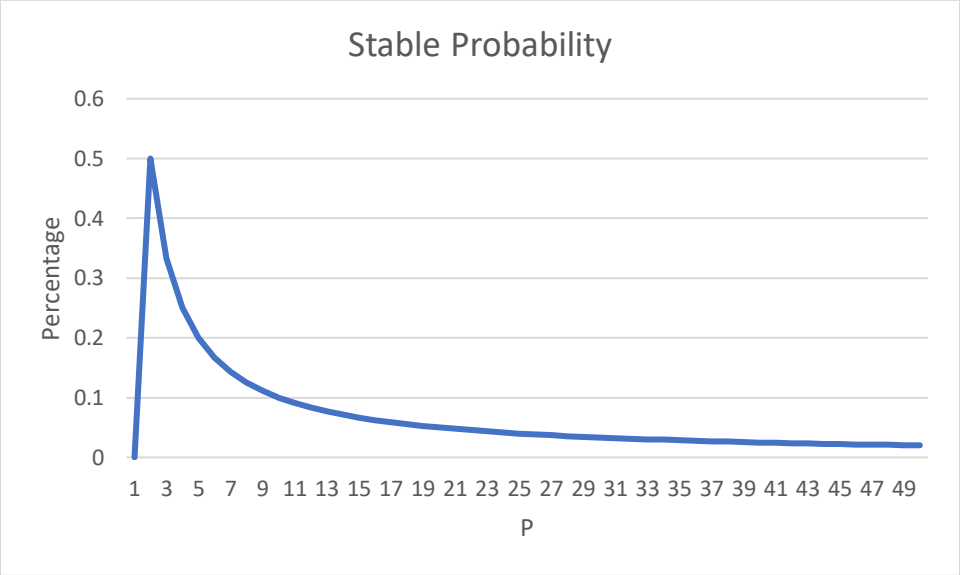
Whether or not an imprint was stable is important, and the record for that is kept and used for creating graphs, which is why it is sent off to a csv.

It should be noted, however, that almost the entirety of what is written above was looped through about five times and then averaged in order to give more accurate numbers.

Results







Analysis

Strangely enough, if the first graph is examined, the reader might find that the program failed to identify any mismatched imprints, which is likely the biggest takeaway from this assignment. Because of this, the program found that each run through of the neural nets found their imprints to be perfect matches and thus stable. It is for this reason that the first graph displays 5 as the number of stable imprints, when the entire thing was looped through only 5

times. This means that there was either an issue with how the program was written, or it is just very good at imprinting. Logically, the former is likely true, and so it seems likely that something was left out, though it is difficult to determine what exactly the issue is, as it would seem that the program was written to the specifications given by the writeup.

Conclusion

As was said before, there was likely an issue with how the imprint was created, or with how it was tested, as the program failed to discover any instances of unstable imprinting. This is a highly unusual outcome, especially when it through most every test. That being said, I felt as if I understood what it was supposed to be doing fairly well. In fact, I am reasonably certain, at least with my current understanding of the topic, that there must be just some small issue embedded somewhere within my code that I cannot seem to find.