Project 3 Report: Hopfield Network Simulation

Patrick Elam

CS420

March 9, 2015

Introduction:

The Hopfield network is a method of simulation an artificial neural network that was invented by John Hopfield in 1982. A Hopfield network simulates arrays of neurons, which can be given a weight, which represents and imprinted memory. Data can then be compared to this network and be “recognized” by the neurons. This is what happens when the neural network stabilizes to its lowest energy state.

Simulator and Calculations:

The simulator I used for this lab was written by myself and Alex Chaloux. We wrote this simulator in C++ to generate and simulate a Hopfield network. To do this, we created 50 sets of patterns, each containing 100 ‘neurons’, or states. We first initialized each value to be either 1 or -1, randomly. We then created a weight network that held a weight associated with each neuron. The weight formula was:

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

After calculating the weights, we tested the neural networks for stability. To do this, we applied the following formulas to create new values for the networks:

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

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

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

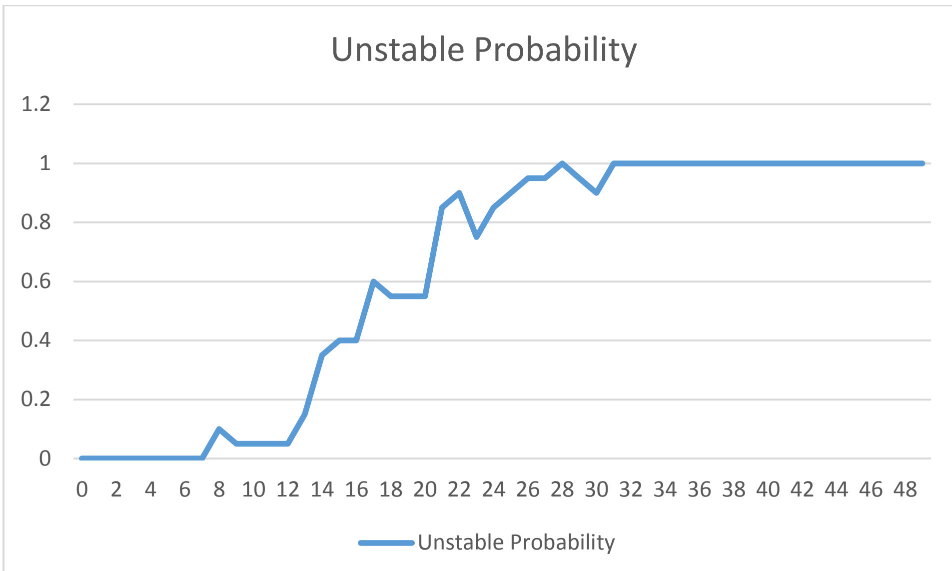
We then compared the old values of the network to the new values. We did this whole process for many iterations, and took the average of how likely a network was to be stable for each P value.

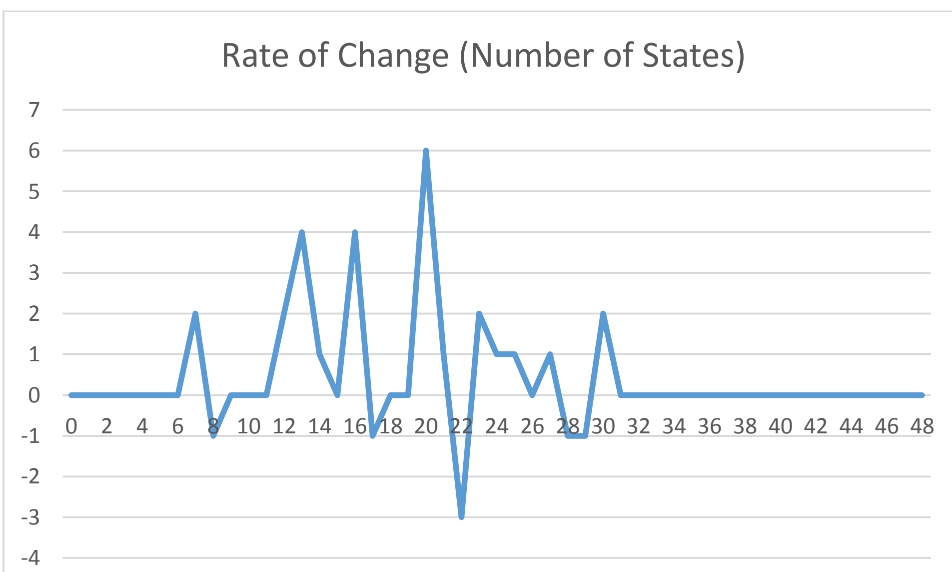
We then graphed the probability of a network to be unstable against its P value, as well as the rate of change of stability.

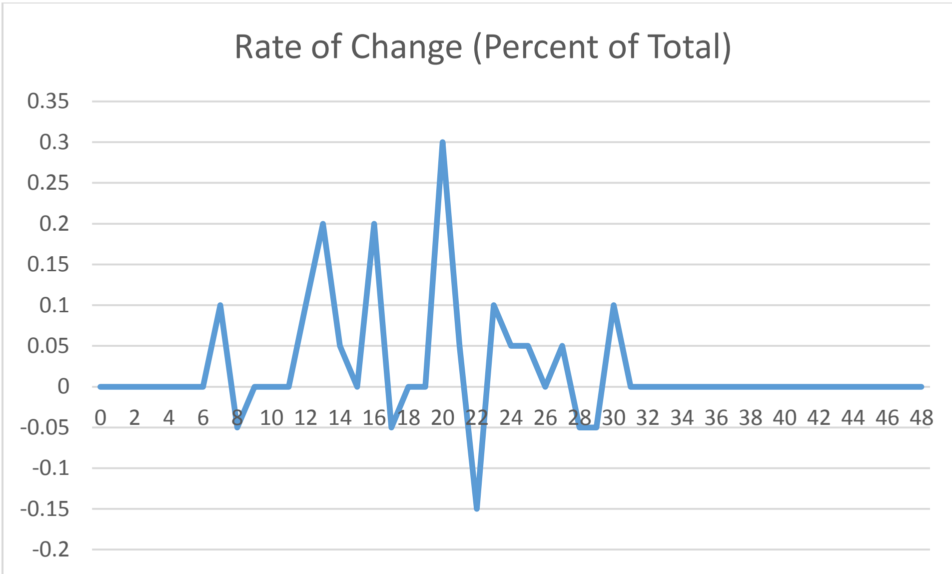
Results:

I ran the simulator three times. Each time, I ran more iterations of the main loop over which the number of stable patterns was averaged. I ran the simulator with 20, 1000, and 10000 iterations, respectively. For each number of iterations, I graphed first the Probability of a Pattern being Unstable vs its P value. I then graphed the Rate of Change of Stable Patterns in number of patterns per P value. I then graphed the Rate of Change of Stable Patterns in percentage of total patterns per P value.

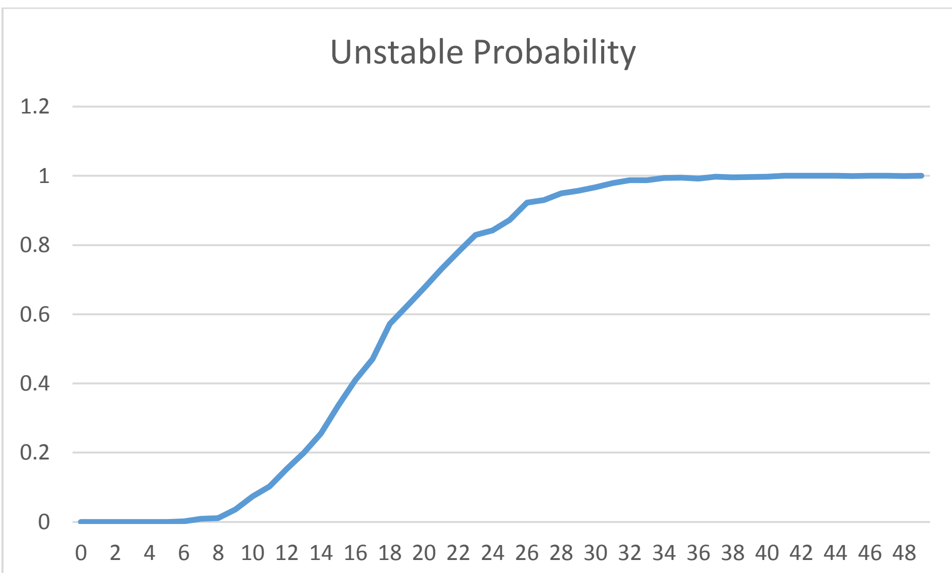
Below are the graphs for 20 iterations:

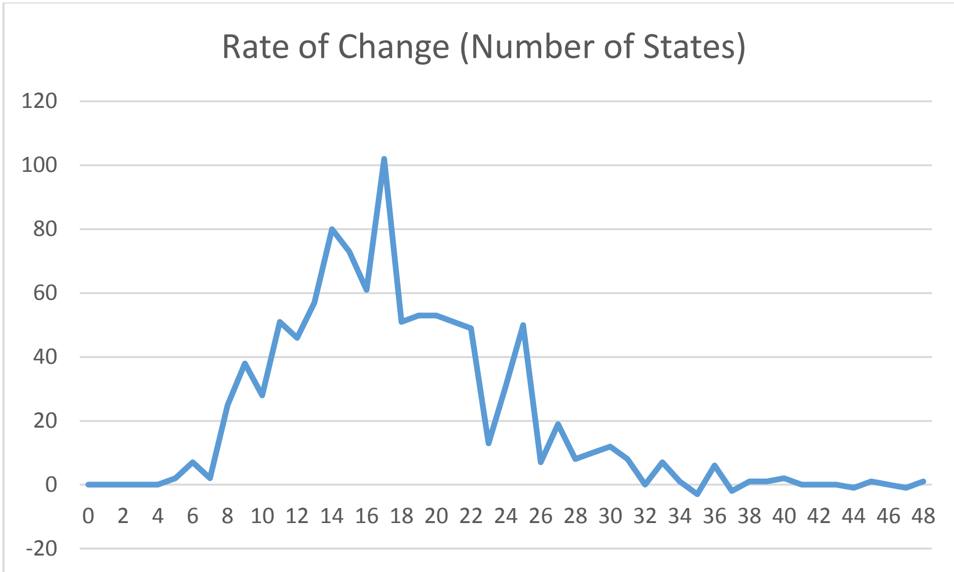


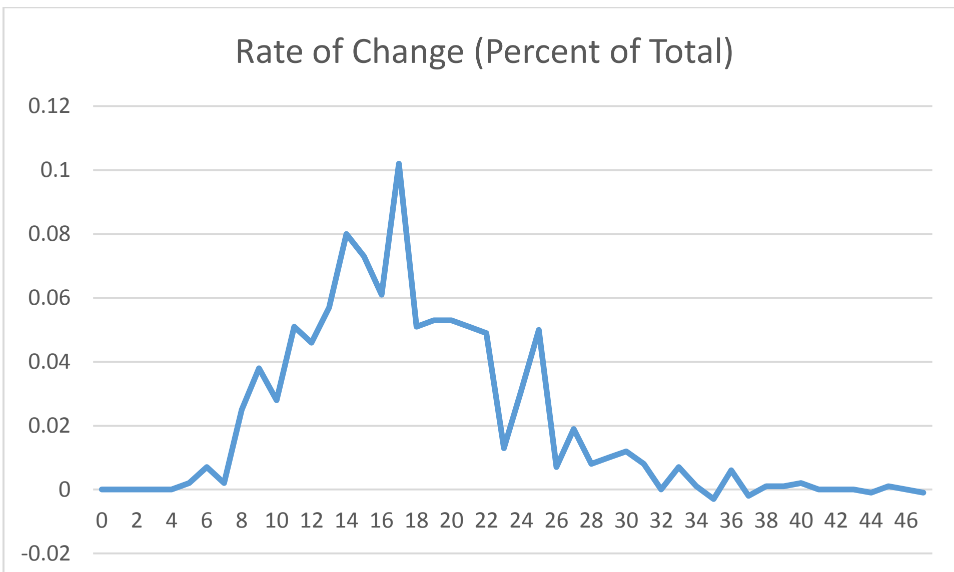




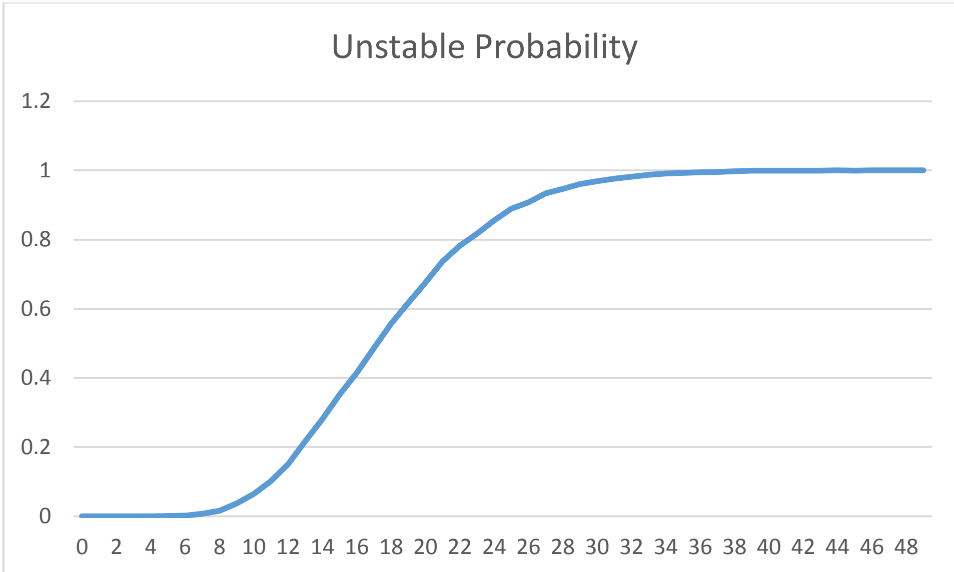
Below are the graphs for 1000 iterations:

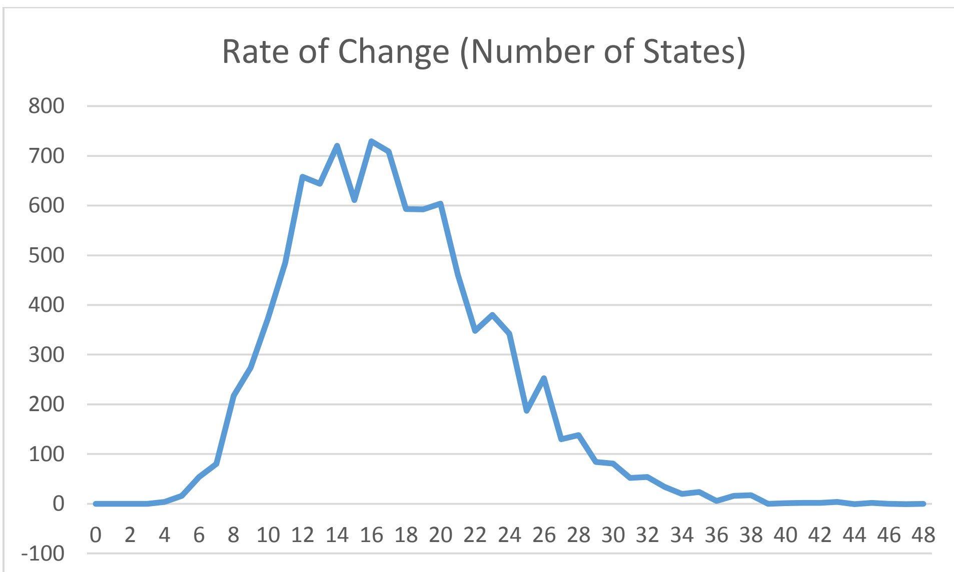


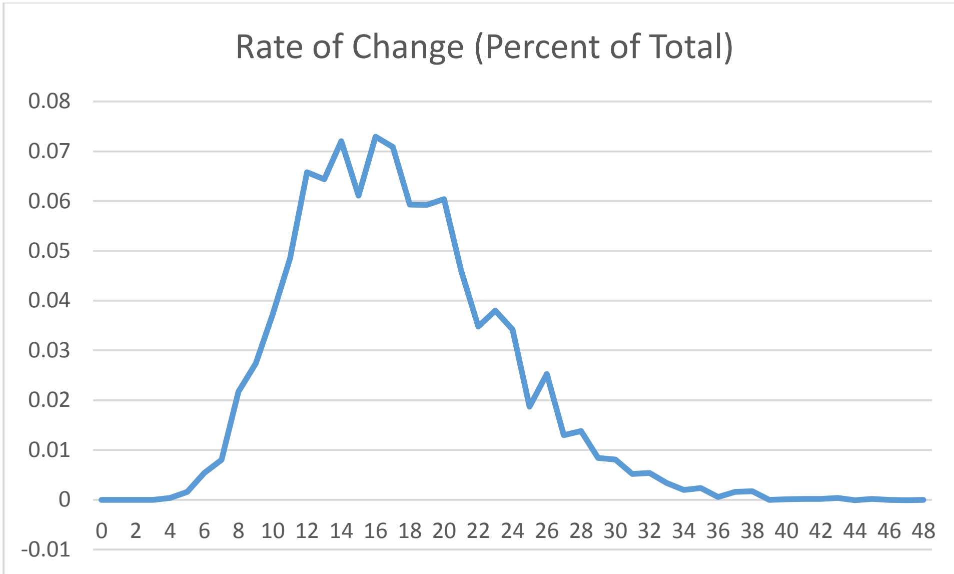




Below are the graphs for 10,000 iterations:


Unstable Probability
1.2
1
0.8
0.6
0.4
0.2
0
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48


Rate of Change (Number of States)
800
700
600
500
400
300
200
100
0
-100
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48



Conclusions:

As is evident from the graphs, as the number of iterations rose, the smoother the lines became. This is because as the number of iterations grew, the less the randomness of the network's initial state impacted the resulting states. It can be seen though that throughout, the shape of the graphs remains constant, although at a lower number of iterations, it is harder to see the shape.

The first graph showed the probability that a given network was unstable when considering the P value of the network. As the patterns approached the 7^{th} or 8^{th} P value, the networks began to become unstable, and became more unstable at an exponentially increasing rate, peaking at about the 16^{th} P value. Onwards the number of unstable networks slowed until about the 40^{th} P value, at which point nearly all of the networks were unstable.

It can be seen that the patterns generated by this simulator are consistent with the behavior that was predicted by Hopfield's theory. The neurons of our artificial network are seen to shift to their lower energy state (as given by their initial weights) and become stable, which can be useful for pattern recognition in computer software.