The Effect of Entropy on Synaptogenic Convergence

Jeffrey Gray

University of Virginia

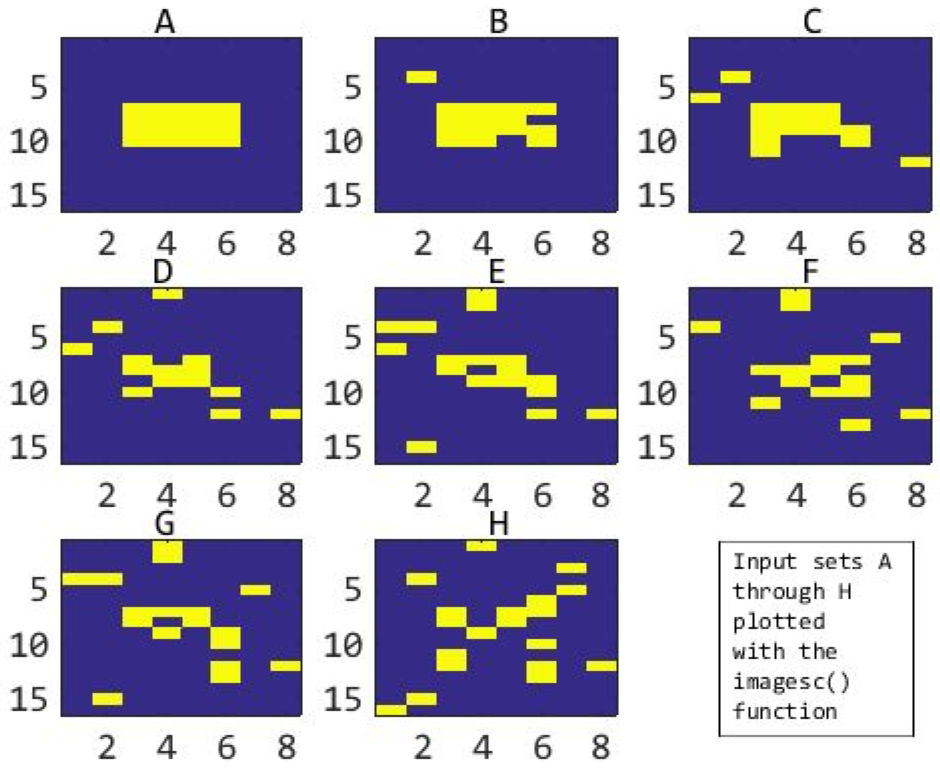
NESC 5330

Introduction

It would place a considerable burden upon our DNA to hard code connectivities between each and every neuron grown in the brain. That’s why we believe this chaotic, unsupervised system relies so heavily upon local modifications and input itself presented into the brain. The growth and modification of these neurons comprise the process of synaptogenesis. Associative synaptic modifications and growth of new neurons may execute at any moment provided that there is enough energy to carry out the metabolic processes. This process represented computationally relies on chaos -- just as it does biologically. I will explore the effect of input entropy on synaptogenic convergence. To do this, I pick a relatively “clean” data set, incrementally inject noise, and view the result. In order to assess the effects of entropy, I use the summary statistics of mutual information (success_val[]) as well as the time required to execute the operation (representative of the algorithm’s complexity).

Implementation

Figure 1: (input_set.mat on GitHub)



The data in A-H represent incremental chaotic permutations (through getNoisyInputs) that apply to the previous set. This noiser function randomly turns on or off one of the input lines within a prototype and is viewable on GitHub. The actual entropy values within the x_chaos vector = [0.4056, 0.4190, 0.4555, 0.4805, 0.4843, 0.5215, 0.5331, 0.5685]. This gradient from order to chaos provides me with a means to determine whether an initially chaotic system (synaptogenesis model at 0 cycles) tends to converge faster towards orderly data or towards data also heavy with entropy.

Making a controller to identify determinants of convergence

In order to run this process, this simple controller() works upstream of runSynaptogenesisModel() in order to populate matrices used for trend analyses.

Figure 2: (from GitHub)

```
function [chaos_val, success_val] = controller(a, b, c, d, e, f, g, h)

   rand = 9711963;

   rng(rand);



   chaos_val = [chaos(a), chaos(b), chaos(c), chaos(d), chaos(e), chaos(f), chaos(g),
chaos(h)];


   success_val = [runSynaptogenesisModel(a), runSynaptogenesisModel(b),
runSynaptogenesisModel(c), ...
       runSynaptogenesisModel(d), runSynaptogenesisModel(e), runSynaptogenesisModel(f), ...

       runSynaptogenesisModel(g), runSynaptogenesisModel(h)]
```

The runtime parameters (a, b, c, d, e, f, g, h) correspond with the input sets in figure 1. The corresponding entropy values and mutual information values are then dumped into chaos_val[] and success_val[], respectively.

Data from Controller

Figure 3:

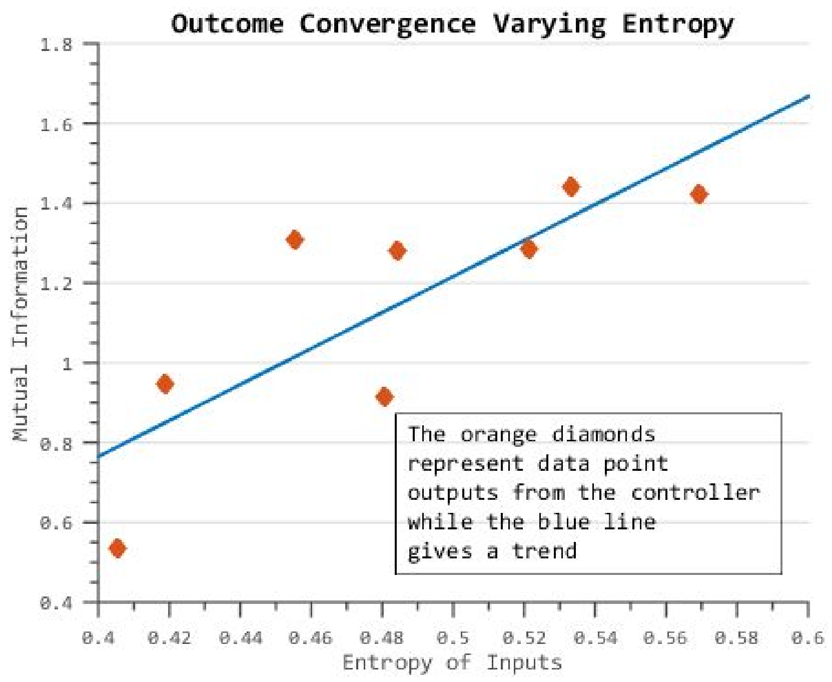
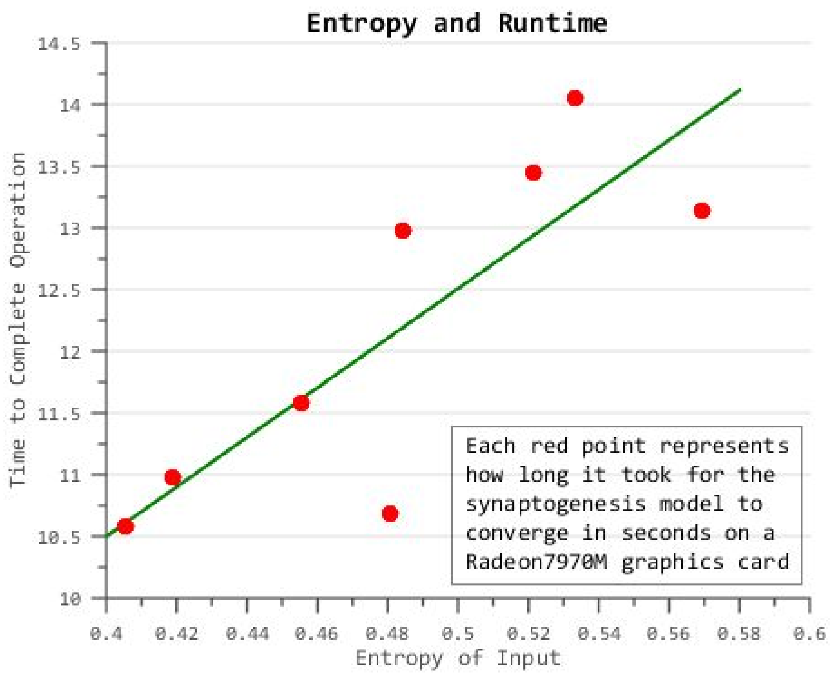


Figure 4:



Interpretation

Contrary to my predictions that the synaptogenesis model would more readily converge to items exhibiting order, the model actually performed the exact opposite. As figure 3 details on the previous page, there is a strong, positive correlation between entropy of the input set and resultant mutual information of the final weight matrix.

While I did not anticipate the synaptogenesis model would behave that way with respect to success of its convergence, I did anticipate the relationship present in figure 4. Even with an incredibly powerful graphics card that I rerouted MATLAB to use instead of my CPU, there still exist considerable disparities between each incremented chaos injection. While the initially chaotic synaptogenesis model basked in the chaos of the input sets and performed well in their presence, it exhibited a much higher algorithmic complexity, as presented by the strong positive correlation.

Given more time, I would love to try and simulate some circuitry, starting with simple logic gates and moving on to higher-order cortical simulations. Specifically, I want to find out how the mPFC serves as a conscious liaison to other brain regions. By finding a way to present a seed with a valid permutations, I could potentially grow some networks and glean insights into how our mPFC could perform these tasks optimally. I would then test the effects of injecting noise into the mPFC at different stages within an attention stream and view which regions receive signals to arborize or increase innervation.