Explaining and Exploring Pattern Completion

Katy Ilonka Gero

December 13, 2018

1 Introduction

Pattern completion is a neural model of how the brain might recall concepts. The basic idea is that every concept is a subset of neurons in a network; I'll call this specific subset of neurons a *pattern*, though there are other names for it like fixed point, attractor, memory, item, word, etc. When the pattern lights up, i.e. the neurons in the pattern fire together, the concept is recalled. However, a common scenario might be that only some of the pattern lights up (e.g. maybe only 50% of the neurons in the pattern; this might be akin to trying to recall someone's name: you have some idea of the name but not enough to say it out loud) and in this case, we'd like to see the structure of the neural network act in such a way as to recover the rest of the neurons in the pattern. We'd like the pattern to complete.

Pattern completion has a long history in neuroscience, mainly starting with Hopfield networks in the 1980s [1, 2], though in this paper I will not be dealing with Hopfield networks. For a good overview of Hopfield networks, along with a Python library to play around with, I recommend *Ch. 17 Memory and Attractor Dynamics* of the textbook Neuronal Dynamics[1].

*

My goal with this paper is twofold. One, I'd like to clearly explain a pattern completion simulation, modeled on the simulation in *Synaptic mechanisms of pattern completion in the hippocampal CA3 network* by Guzman, Schlogl, Frotscher, and Jonas [3], such that along with the Python Jupyter notebooks found in the github repository[2] accompanying this paper one can quickly and easily replicate (and ideally improve upon) the results found here. Two, I'd like to share some results on how pattern completion varies when the underlying neural connectivity matrix is not randomly connected. These results are from my simulations and, like the simulation model itself, similarly modeled after those in Guzman et. al.

In this paper, I assume little to no prior knowledge of neuroscience or graph theory; I assume moderate comfort with matrix operations. To follow along with the code, you should be familiar with Python, numpy, and scipy's implementation of sparse matrices. Note that I include all papers cited in this work in the github repository.

[1] `https://neuronaldynamics.epfl.ch/online/Ch17.html`

[2] `https://github.com/kgero/pattern-completion`

2 Definition of model

My understanding of the Guzman et. al. model comes from their supplementary materials[3]. Guzman et. al. are modeling the hippocampal CA3 network, which plays a key role in learning and memory. Their simulations follow previous work, in which the network has the following properties: binary neurons, recurrent excitatory connectivity, clipped Hebbian plasticity, global threshold, linear global inhibition, and iterative recall.

These properties are made clear by walking through the computational steps of running the simulation. For the most part, I follow the notation in Guzman et. al., and indicate where I break from it:

(a) **Define a connectivity matrix W.** This n x n matrix defines the connections between all neurons in the network. All connections, i.e. entries in the matrix, are either 0 (not connected) or 1 (connected). e.g. $W_{ij} = 1$ indicates that neurons i and j are connected. In the base case, it is randomly connected given some probability p, typically on the order of $p = 0.03$ or 3% connectivity.

(b) **Generate m patterns $\{p_1, ..., p_m\}$.** Each pattern is a binary n-vector representing which subset of neurons are activated for the given pattern. An average activity level f (assumed to be 0.001 for the entirety of this paper; this number comes from experimental work) indicates how many neurons should on average be active in a given pattern; activated neurons are selected at random.

(c) **Storage of patterns, resulting in synaptic weight matrix J.** This n x n matrix indicates the connections learned based on the patterns stored. Weights are updated using a clipped Hebbian rule: at first all connections are 0, then if neuron i and neuron j are active in the same pattern, $J_{ij} = J_{ji} = 1$. In this way, we 'load' all the patterns into matrix J and then clip all values at 1. Mathematically, this amounts to

$$J = \sum_{p=1}^{m} p \cdot p^T \tag{1}$$

where all entries are clipped to the maximum value of 1 and $\cdot$ represents the dot product. This model is explained in clear and extensive detail in *Dynamics of the CA3 pyramidal neuron autoassociative memory network in the hippocampus* [4]. Note that while W is not symmetric, J is symmetric.

(d) **Mask J with W.** W represents the possible connections in the network, thus we need to remove any connections from the learning not 'possible'. This amounts to element-wise multiplication of W and J, which I indicate as J_W. (Note that this breaks with the Guzman et. al. supplementary material notation, which keeps W and J separate and multiplies them when necessary.)

[3] `http://science.sciencemag.org/content/suppl/2016/09/07/353.6304.1117.DC1`

(e) **Degrade patterns.** In order to see if the network can recall patterns, we need degraded patterns to 'trigger' the network. To degrade the patterns, we remove some fraction b_{valid} of neurons originally activated in the pattern and add some fraction b_{spurious} of neurons originally not activated in the pattern. I follow Guzman et. al. and set $b_{\text{valid}} = 0.5$ and $b_{\text{spurious}} = 0.001$.

(f) **Simulate iterative recall triggered by degraded patterns.** Finally, we simulate recall by inputting a degraded pattern into the network, which results in a new set of activated neurons. Let's set up some notation:

- X_t is the state (the binary n-vector of which neurons are activated at time t),
- g_1 is an inhibition factor, and
- g_0 is an activation threshold.

We calculate X_{t+1}, the new set of activated neurons, given X_t, the set of input activated neurons, with the following expression:

$$X_{t+1} = [\frac{1}{n} J_W \cdot X_t] - [\frac{1}{n} g_1 X_t] > g_0 \tag{2}$$

Note that the right side of this expression is a boolean, and hence the resulting state X_{t+1} will also be a binary n-vector.

We repeat this calculation for several time steps, until $X_{t+1} = X_t$ in the steady state. This typically happens within 10 time steps, though not always. (Note that equation 2 differs slightly from Guzman et. al., as they include an extra matrix P to take into account stochastic variation in transmitter releases. Refer to page 11 of their supplementary materials.)

Guzman et. al. set $g_0 = 7 * 10^{-6}$, corresponding to 3 excitatory inputs required to activate a neuron for a network size of 330,000 neurons. Thus, I set $g_0 = 3/n$. Guzman et. al. vary g_1 between 0 and 0.05; for the work in this paper, I set $g_1 = 0$. Given this, equation 2 simplifies to:

$$X_{t+1} = J_W \cdot X_t > 3 \tag{3}$$

3 Including motifs

The main thrust of Guzman et. al. is that the neurons in the hippocampus are not randomly connected; in the simulation this amounts to saying that W is not random. Instead, they experimentally found certain connectivity motifs to occur more often than expected in a randomly connected graph. In *Synchronization from second order network connectivity*

statistics by Zhao, Beverlin, Netoff, and Nykamp [5] they outline how to generate the four main connectivity motifs.

Figure 1: From Zhao et. al. [5]

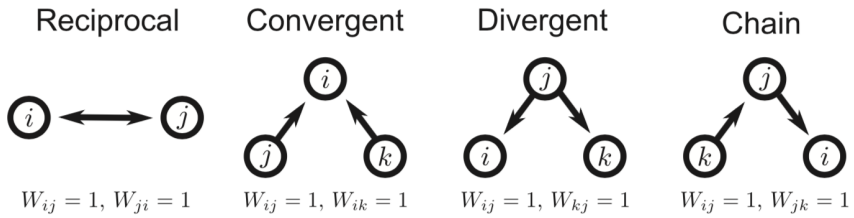


FIGURE 1 |The four second order connection motifs of reciprocal, convergent, divergent, and chain connections.

However, in this work I increase the motifs using a simpler, though less precise, method. I select a set of core 'preferred' neurons and create a disproportionate number of connections within this core set. By selecting some fraction of neurons *core* to be preferred, and some fraction of connections *split* > *core* to be reserved for the core, I can increase the number of connectivity matrix. (Note that if *split* = *core* = 0, this results in the normal, randomly connected matrix.)

This allows me to increase the number of connectivity motifs, as the core neurons are more likely connected to each other. I don't have the same control as a more explicit algorithm would, but it's quick and simple.

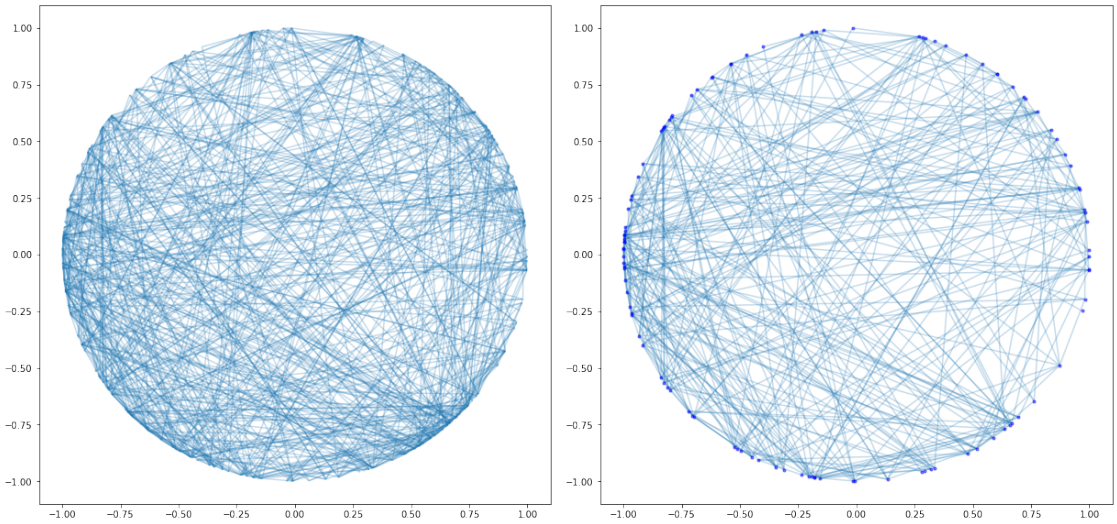
4 Visualizing the simulation in the base case

In the base case, W is randomly connected. For this walkthrough, I'll use the following parameters:

$n = 100,000$	number of neurons
$p = 0.03$	connectivity of W
$m = 2$	number of patterns
$f = 0.001$	average activation of each pattern

Below, I've graphed the connectivity matrix in two cases. In each case, all neurons are placed on the edge of a circle, and activated connections are drawn as lines between the neurons. On the left, the connections in the final connectivity matrix J_W are plotted. On the right, the first pattern, p_1, is plotted as points along the circle of the activated neurons, and the connections this pattern activates in J_W as lines between neurons.

(a) J_W (b) $J_W \cdot p_1$



At each time step I calculate the correlation between the input activation and initial pattern as

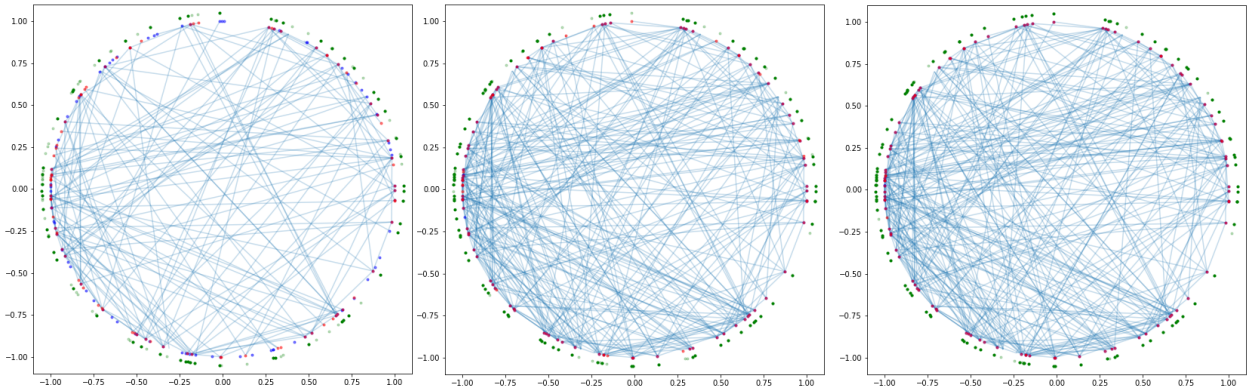
$$\text{corr}_t = 1 - \text{cosine-dist}(p_{\text{init}} - \frac{\sum_{i=0}^{n} p_{\text{init},i}}{n}, X_t - \frac{\sum_{i=0}^{n} X_{t,i}}{n}) \tag{4}$$

such that when $p_{\text{init}} = X_t$ then the correlation is 1, and any difference, including more or less activated neurons, results in a correlation of less than 1.

Finally, I create a set of plots for each time step in the simulation. The original pattern is plotted in green at the outside of the circle; light green indicates a neuron in the original pattern not currently active in the network. Full pattern completion is achieved when all green points are dark, and there are no spurious connections.

The input activation is plotted in blue at the edge, and the output activation is plotted in red. The connections that the input activation activates are drawn as connecting lines. Note that these connections do not necessarily activate every neuron they touch, given the threshold in equation 3. Below I show three time steps, with the first input activation as a degraded version of pattern 1.

(a) $t = 0, \text{corr} = 0.51$ (b) $t = 1, \text{corr} = 0.88$ (c) $t = 2, \text{corr} = 0.95$



Note that in the first time step, there are many red and blue points along the edge of the circle; this indicates that the network is not very stable as many input activations will not be activated at the next step. One can also see that several blue points (input activations) have no connections, indicating there are no connections in J_W for them to activate. (Conversely, there are no red points without connections.) By $t = 2$, almost all points are purple: both blue and red, indicating a stable neuron that is activated both at the input and output.

By examining the green points, we can see that in the first time step there are many activated neurons not in the pattern; at 100,000 neurons, even a small fraction of spurious activations can be a large number. At each time step, less and less spurious neurons are activated; however in these three steps we do not see all neurons in the pattern activated, hence the correlation of 0.95.

5 Simulation results

I run all my simulations with 100,000 neurons. This is approximately 1/3 of the size of the main simulations in Guzman et. al., though they do have a simulation with 110,000 neurons in which they find that pattern completion is much reduced and the inclusion of motifs do not adequately recover pattern completion. However, 100,000 neurons is the size of simulations I could easily run on my computer without modifications and, in general terms, is quite large.

First I show the increase in three of the four motifs with different settings of the size of the core and the number of connections reserved for the core. (I don't calculate the number of chain motifs simply as I ran out of time.)

Table 1: Increase in motifs due to preferred core neuron connectivity. *Split* is the fraction of connections to reserve for the core group of neurons. *Core* is the fraction of neurons in the core group.

Split	Core	# reciprocal (incr.)	# convergent (incr.)	# divergent (incr.)
0.0	0.0	500 thousand (1x)	50 trillion (1x)	50 trillion (1x)
0.9	0.4	2.5 million (4.9x)	110 trillion (2.2x)	110 trillion (2.2x)
0.9	0.3	4.2 million (8.3x)	144 trillion (2.9x)	144 trillion (2.9x)
0.9	0.2	8.2 million (16x)	212 trillion (4.2x)	212 trillion (4.2x)
0.9	0.1	17.6 million (35x)	414 trillion (8.3x)	414 trillion (8.3x)

Then I run simulations with a variety of values of p (the connectivity of the base matrix W) and m (the number of patterns learned), in two cases: W is randomly connected, and W has an increased number of connectivity motifs (*split=0.9* and *core = 0.4*).

For each run, I input a degraded version of the first pattern ($b_{\text{valid}} = 0.5$ and $b_{\text{spurious}} = 0.001$) and run the simulation for 10 time steps, reporting the correlation of the state at the 10th time step with the original pattern. Each value in the heat map is the average final correlation of 10 runs of the simulation.

(a) W is random. (b) Incl. motifs. (*0.9, 0.4*) (c) Difference.

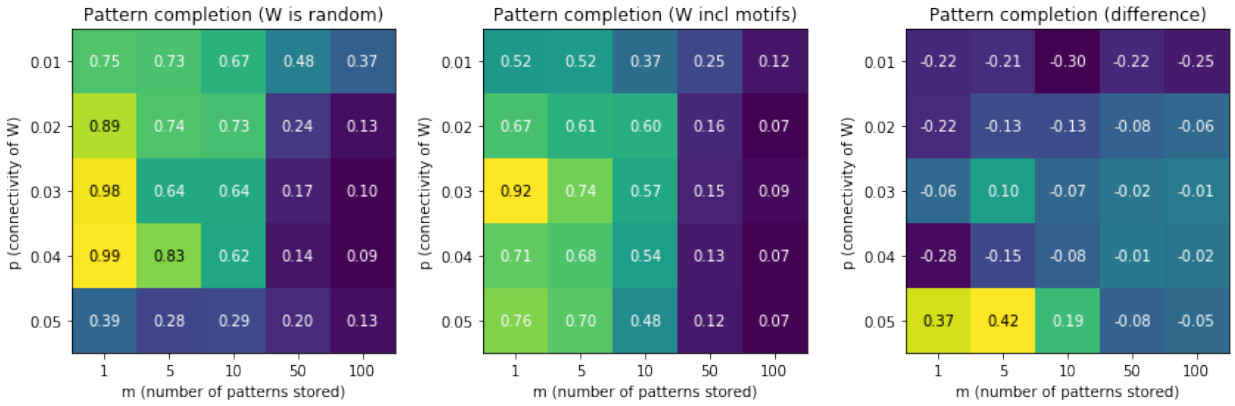


Figure 4: Pattern completion across values of p and m in two cases: W is randomly connected, and W contains an increased number of motifs. Fig (c) shows the difference: positive values indicate that increasing the number of motifs *improved* pattern completion.

Pattern completion is most robust at $p = 0.04$ when W is random and $p = 0.03$ when motifs are increased.

As in Guzman et. al., I found that the introduction of motifs did not always improve pattern completion; in many cases it hurt pattern completion. However, there were some configurations in which it helped, notably $p = 0.05; m = 5$ in which case the motifs over doubled the pattern completion ability.

It's also interesting to note the effect of the connectivity level, p, on pattern completion. While m has a clear relationship – more patterns decreases pattern completion in all cases – p seems to peak at certain values, different with the inclusion of extra motifs or not.

I believe that it is also highly dependent on the size of the network. At 100,000 neurons, pattern completion is much less than at 330,000 neurons, which is consistent with the results in Guzman et. al. In Fig. S11 of their supplementary materials they show pattern completion of 110,000 neurons at $p = 0.03$ to be at about 0.8 for $m = 5$; I find it to be 0.6. In their simulations, with a network size of 330,000 neurons and $p = 0.03$, they were able to store and successfully complete almost 10,000 patterns; in my smaller simulation I could barely reach 10 patterns in any configuration.

6 Conclusion

My simulations show that increasing the number of certain connectivity motifs does have an effect on pattern completion, though this effect is not always positive, which is consistent with results in Guzman et. al. Further work could look into why pattern completion seems so hampered at this size (100,000 neurons), or what changes to the connectivity matrix could recover pattern completion of networks of this size. Hopefully I have succeed in my first goal of clearly explaining the simulation of pattern completion I performed, and releasing the code in an accessible way.

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[2] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

[3] S. J. Guzman, A. Schlögl, M. Frotscher, and P. Jonas, "Synaptic mechanisms of pattern completion in the hippocampal ca3 network," *Science*, vol. 353, no. 6304, pp. 1117–1123, 2016.

[4] M. R. Bennett, W. G. Gibson, and J. Robinson, "Dynamics of the ca3 pyramidal neuron autoassociative memory network in the hippocampus," *Philosophical Transactions: Biological Sciences*, vol. 343, no. 1304, pp. 167–187, 1994.

[5] L. Zhao, B. Beverlin, T. Netoff, and D. Nykamp, "Synchronization from second order network connectivity statistics," *Frontiers in Computational Neuroscience*, vol. 5, p. 28, 2011.