Towards Self-Organizing Epsilon Machines:
Measuring the Organization in Self-Organizing Maps

Jules Litman-Cleper James Crutchfield , Mikhael Semaan
PHY 256B, Spring 2022

Introduction

Part I: Measuring the Organization in Self-Organizing Maps (SOMs)

- Motivation
- Initial Channel Capacity
- Converged Channel Capacity
- Summary Interpretation

Part II: Extending the Self-Organizing Map for the reversal Process (SOeM)

- Model Architecture

Part III: Discussion

- Technical Limitations
- Biological Plausibility: mirror representations as possible reversal processes
- Conceptual Significance: Emergence of Causal Inference

Introduction

The Self-Organizing map (SOM) is a compelling example of a relatively simple (single layer) computational process that can derive its own structure from input data. The structure derived is a topology preserving, PCA equivalent, dimensionality-reducing, 'similarity' clustering of the inputs. SOMs were designed by Tuevo Kohonen in 1982 for the explicit purpose of capturing the dynamics that result in the topographically organized brain maps found in the cortices of mammals. Kohonen also showed that these maps can discover "representations" or "internal structures" from the input data such as the semantics within sentences. Still we do not fully understand how brain maps, and their topological organization, relate to inferential processes like predictive learning, and how these biological inferential dynamics emerged out of evolutionary processes.

Epsilon Machines (eMs) are the unique, minimal and lowest computational level models that can optimally predict a set of data or natural process. More directly, eMs group together past states that predict future states and do so using a reversal process that retrodicts the past and a forward process that predicts future states based on groupings of pasts. Because the eM can make predictions about future observations given past observations, for our purposes here, eMs are a way of building causal inferences, or at least an exemplar of the basics of how causal mappings can be formed. In being the lowest computational power and the most minimal model, they also align to biological constraint optimization such as using the most minimal amount of resources while still achieving predictive learning.

Since the self-organizing map (SOM) is a "pattern classifier" in which the responsiveness of certain cells/nodes are grouped as subsets that correspond with a discrete class of patterns, the resemblance to epsilon Machines (eMs) already intrigues analysis. Could a similar mechanism in the SOM be applied to induce a process that generates eMs from data, in a more unsupervised manner than the eM reconstruction algorithm?

Part I: Measuring the Organization in Self-Organizing Maps (SOMs)

Motivation

Firstly is to measure how Shannon information changes in the initial and converged learning epochs of the SOM, to understand the dynamics of information processing as the SOM learns. We will consider the SOM as a channel and look at channel capacity. This means that instead of a stream of an alphabet through a channel we are looking at a stream of the nodes that have become the "best matching unit." SOMs use both competitive learning (best-matching unit wins) and collaborative learning (neighborhood radius function).

In thinking about the SOM as a channel we also need to consider the neighborhood radius function, which updates the closest neighbors of the best matching unit with a parameterized step size and decay amount. The channel capacity will be very different when we change the amount of collaboration and competition via the radius factor. With the neighborhood radius function the 'channel' will become more complex- which is to say different probabilities are at play. Understanding the information architecture of these two tradeoffs (competition/cooperation) is already profound in the sense that these are information-architectural dynamics which can be found biologically, both among neuronal populations as well as in evolutionary population dynamics. The shared presence of these dynamics suggest a possible information-process that is a signature of self-similarity contained within these two nested and strikingly innovative natural processes.

To be followed up in future studies, we will suggest ways to look at the structures of information that arise out of the separate competitive/cooperative learning dynamics by analyzing how the entropy changes as you change the neighborhood radius function in the map. This neighborhood radius function could be modeled as a control parameter in a dynamical system- to consider spaces of dynamical systems and what kinds might be candidates for the emergence of eMs out of naturally organizing systems. Perhaps competitive vs. collaborative learning are each a kind of dynamical system whose interaction organizes information and complexity. Perhaps competitive and cooperative dynamics sustain a kind of 'edge of chaos' criticality that is necessary for learning in neural networks more generally[1] and for learning processes with information in natural systems (speculation). For now let us return to the channel capacity of the self-organizing map. We will use the Even process in the input and the Faircoin in the initial weights, so that in theory, the SOM will need to add a state to be able to adequately

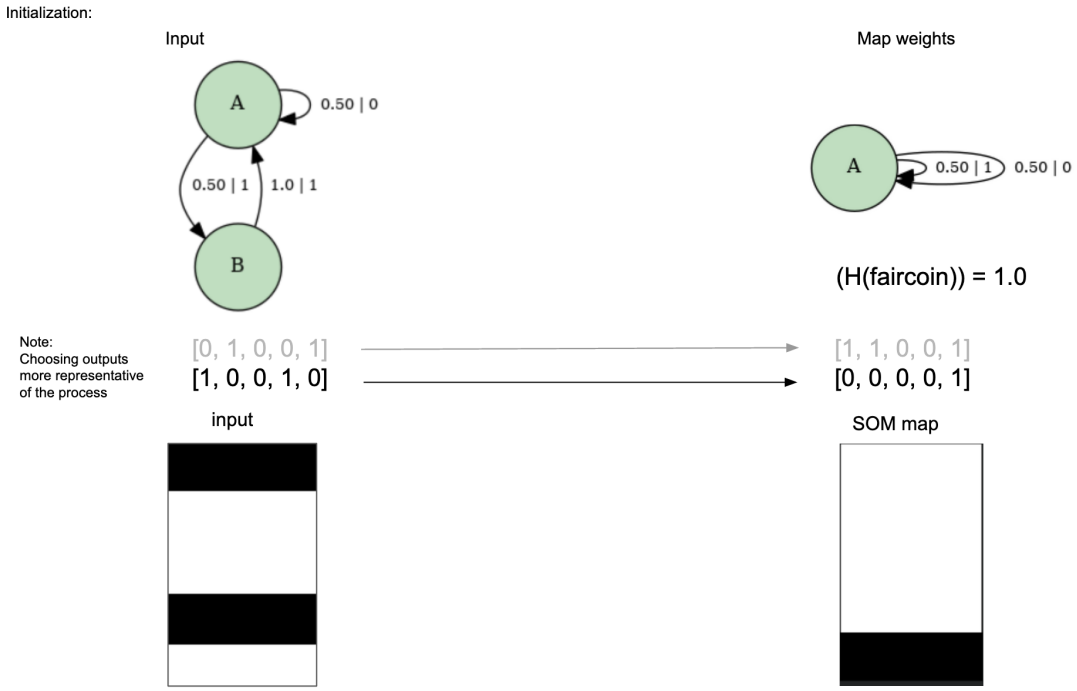
[1] See "Deep Information Propagtion"
https://www.semanticscholar.org/paper/Deep-Information-Propagation-Schoenholz-Gilmer/4fdc7df2c737141a1bf5aec27a438b77d01f8af0

map the inputs. By thinking of the SOM as an information channel we can measure the equivalency, ambiguity and channel capacity and see how this changes over learning epochs with different neighborhood parameters.

(a) Initial Channel Capacity

1. generate a *random* set of binary values, this "randomness" will specifically be generated out of an already known eM process, the Even Process, so that the information measures are known.

2. initialize a one dimension SOM with 5 nodes, and set their weights with another version of known randomness, the Fair Coin Process.



Channel Capacity at Initial State:

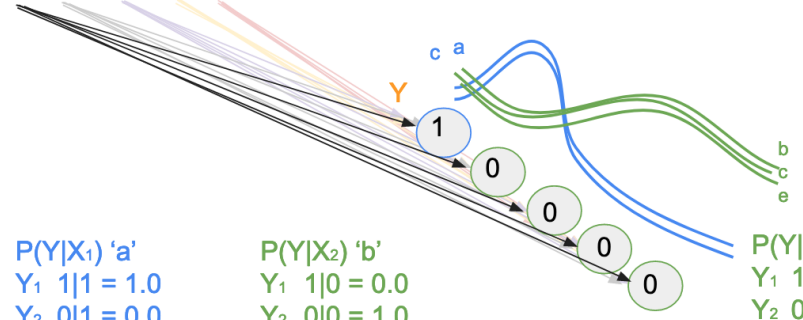
a. To figure out a probability distribution we need for each chosen learning epoch:
   i. Probability of activation of a node:

$$P_{\text{learning epoch}}(\text{map}|\text{input})$$

Note: I believe here the P(Y|X) will be the same as P(X|Y)? Also for the neighborhood radius, adjacent probabilities are set to 0.5 but this could be set differently, see technical limitations.

Input vector

X_1 X_2 X_3 X_4 X_5
1 0 0 1 0



Probability of node $Y_{1\text{-}5}$ being a "Best Matching Unit" aka weight will get updated

$P_0(Y|X) = 0.112$ at initial state which is learning epoch 0 denoted by P_0

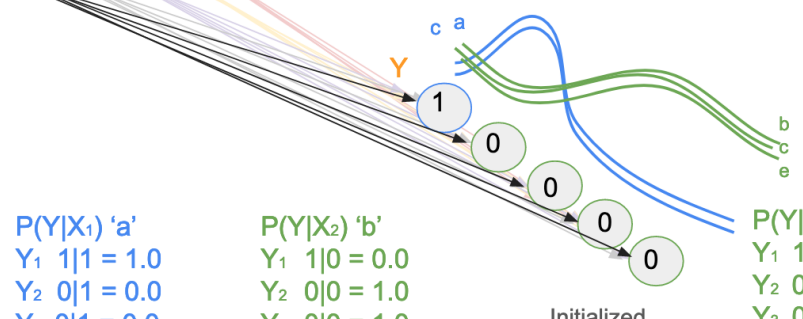
Probability distributions
For each input value X1-X5 then these get added up to a total average of 0.2+0.8+0.8+0.2+0.8 = 2.8/25 = 0.112

Initialized random weights of 1D Map (Y)

	$P(Y\|X_1)$ 'a'	$P(Y\|X_2)$ 'b'	$P(Y\|X_3)$ 'c'	$P(Y\|X_4)$ 'd'	$P(Y\|X_5)$ 'e'
1	Y_1 1|1 = 1.0	Y_1 1|0 = 0.0	Y_1 1|0 = 0.0	Y_1 1|1 = 1.0	Y_1 1|0 = 0.0
2	Y_2 0|1 = 0.0	Y_2 0|0 = 1.0	Y_2 0|0 = 1.0	Y_2 0|1 = 0.0	Y_2 0|0 = 1.0
3	Y_3 0|1 = 0.0	Y_3 0|0 = 1.0	Y_3 0|0 = 1.0	Y_3 0|1 = 0.0	Y_3 0|0 = 1.0
4	Y_4 0|1 = 0.0	Y_4 0|0 = 1.0	Y_4 0|0 = 1.0	Y_4 0|1 = 0.0	Y_4 0|0 = 1.0
5	Y_5 0|1 = 0.0	Y_5 0|0 = 1.0	Y_5 0|0 = 1.0	Y_5 0|1 = 0.0	Y_5 0|0 = 1.0
	Since only 1 unit is the likely active one Out of 5 nodes	Since 4 units are the likely active Out of 5 nodes	Since 4 units are the likely active Out of 5 nodes	Since only 1 unit is the likely active one Out of 5 nodes	Since 4 units are the likely active Out of 5 nodes
	$P(Y\|X_1)$ = ⅕ = 0.2	$P(Y\|X_2)$ = ⅘ = 0.8	$P(Y\|X_2)$ = ⅘ = 0.8	$P(Y\|X_1)$ = ⅕ = 0.2	$P(Y\|X_2)$ = ⅘ = 0.8

Input vector

X_1 X_2 X_3 X_4 X_5
1 0 0 1 0



Probability of node $Y_{1\text{-}5}$ being a "Best Matching Unit" aka weight will get updated

$P_0(Y|X) = 0.112$ at initial state which is learning epoch 0 denoted by P_0

Probability distributions
For each input value X1-X5 then these get added up to a total average of 0.2+0.8+0.8+0.2+0.8 = 2.8/25 = 0.112

Initialized random weights of 1D Map (Y)

	$P(Y\|X_1)$ 'a'	$P(Y\|X_2)$ 'b'	$P(Y\|X_3)$ 'c'	$P(Y\|X_4)$ 'd'	$P(Y\|X_5)$ 'e'
	Y_1 1|1 = 1.0	Y_1 1|0 = 0.0	Y_1 1|0 = 0.0	Y_1 1|1 = 1.0	Y_1 1|0 = 0.0
	Y_2 0|1 = 0.0	Y_2 0|0 = 1.0	Y_2 0|0 = 1.0	Y_2 0|1 = 0.0	Y_2 0|0 = 1.0
	Y_3 0|1 = 0.0	Y_3 0|0 = 1.0	Y_3 0|0 = 1.0	Y_3 0|1 = 0.0	Y_3 0|0 = 1.0
	Y_4 0|1 = 0.0	Y_4 0|0 = 1.0	Y_4 0|0 = 1.0	Y_4 0|1 = 0.0	Y_4 0|0 = 1.0
	Y_5 0|1 = 0.0	Y_5 0|0 = 1.0	Y_5 0|0 = 1.0	Y_5 0|1 = 0.0	Y_5 0|0 = 1.0
	Since only 1 unit is the likely active one Out of 5 nodes	Since 4 units are the likely active Out of 5 nodes	Since 4 units are the likely active Out of 5 nodes	Since only 1 unit is the likely active one Out of 5 nodes	Since 4 units are the likely active Out of 5 nodes
	$P(Y\|X_1)$ = ⅕ = 0.2	$P(Y\|X_2)$ = ⅘ = 0.8	$P(Y\|X_2)$ = ⅘ = 0.8	$P(Y\|X_1)$ = ⅕ = 0.2	$P(Y\|X_2)$ = ⅘ = 0.8

ii. Here is the *equivocation* H[Y|X], *ambiguity*: H[X|Y], and mutual information I[X:Y] for initial state (learning epoch 0) with no neighborhood radius. In this case we use the activated node probability divided by the total number of activated nodes, which was 14 out of the 25 total samples, the rest being 0.

Initial State Channel Capacity

neighborhood parameter = 0

measure	bits
H[X|Y]	1.714
H[Y|X]	1.501
I[X:Y]	0.592

```
Channel ambiguity 1.7142857142857153
Channel equivocation 1.5013964291895632
BMU channel capacity 1.0
Maximizing input source:
Class: Distribution Alphabet:
('1', '2', '3', '4', '5') for all rvs Base:
linear Outcome Class: str Outcome Length: 1
RV Names: ('X',)
x p(x)
1 1/2
2 1/8
3 1/8
4 1/8
5 1/8
```

neighborhood parameter = 1

measure	bits
H[X|Y]	2.033
H[Y|X]	1.884
I[X:Y]	0.278

```
Channel ambiguity 2.033265152826344
Channel equivocation 1.8842669535570917
BMU channel capacity 0.38147926435258356
Maximizing input source:
Class: Distribution Alphabet:
('1', '2', '3', '4', '5') for all rvs Base:
linear Outcome Class: str Outcome Length: 1
RV Names: ('X',)
x p(x)
1 2595/6382
2 1/4281385
3 6461/32665
4 6461/32665
5 6461/32665
```

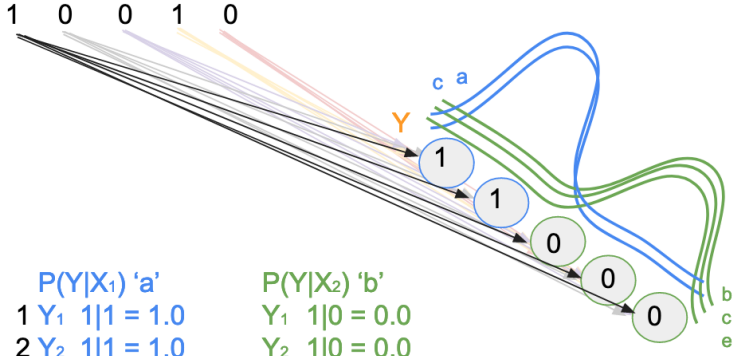
Interpretation: The translated probabilities into dit is possibly incorrect (see discussion: technical limitations, for details on this). However here it could be that things are still noisy/disorganized, there is high ambiguity and equivocation, but there is some mutual information between the input and the mapping because the weights are dependent on the inputs and are likely to activate/update as a function of the input. These probabilities assume there is no neighborhood radius function.

Channel Capacity at convergence

a. Channel capacity at convergence (epoch 10) without and then with a neighborhood parameter:

Input vector

X_1 X_2 X_3 X_4 X_5
1 0 0 1 0



Converged weights of 1D Map (Y)

Probability of node $Y_{1\text{-}5}$ being a "Best Matching Unit" aka weight will get updated

$P_{10}(Y|X)$ = 0.104 at converged state, learning epoch 10

Probability distributions
For each input value X1-X5 then these get added up to a total average of 0.4+0.6+0.6+0.4+0.6 = 2.6/25 = 0.104

$P(Y|X_1)$ 'a'
1 Y_1 1|1 = 1.0
2 Y_2 1|1 = 1.0
3 Y_3 0|1 = 0.0
4 Y_4 0|1 = 0.0
5 Y_5 0|1 = 0.0

Since only 1 unit is the likely active one Out of 5 nodes

$P(Y|X_1)$ = ⅖ = 0.4

$P(Y|X_2)$ 'b'
Y_1 1|0 = 0.0
Y_2 1|0 = 0.0
Y_3 0|0 = 1.0
Y_4 0|0 = 1.0
Y_5 0|0 = 1.0

Since 4 units are the likely active Out of 5 nodes

$P(Y|X_2)$ = ⅗ = 0.6

$P(Y|X_3)$ 'c'
Y_1 1|0 = 0.0
Y_2 1|0 = 0.0
Y_3 0|0 = 1.0
Y_4 0|0 = 1.0
Y_5 0|0 = 1.0

Since 4 units are the likely active Out of 5 nodes

$P(Y|X_2)$ = ⅗ = 0.6

$P(Y|X_4)$ 'd'
Y_1 1|1 = 1.0
Y_2 1|1 = 1.0
Y_3 0|1 = 0.0
Y_4 0|1 = 0.0
Y_5 0|1 = 0.0

Since only 1 unit is the likely active one Out of 5 nodes

$P(Y|X_1)$ = ⅖ = 0.4

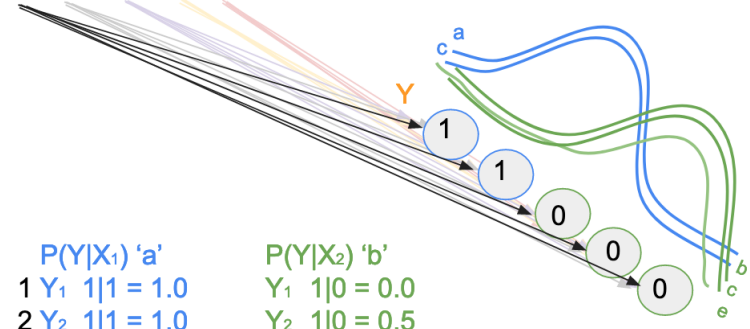
$P(Y|X_5)$ 'e'
Y_1 1|0 = 0.0
Y_2 1|0 = 0.0
Y_3 0|0 = 1.0
Y_4 0|0 = 1.0
Y_5 0|0 = 1.0

Since 4 units are the likely active Out of 5 nodes

$P(Y|X_2)$ = ⅗ = 0.6

Input vector

X_1 X_2 X_3 X_4 X_5
1 0 0 1 0



Converged weights of 1D Map (Y)

Probability of node $Y_{1\text{-}5}$ being a "Best Matching Unit" aka weight will get updated
with a 1-step neighborhood 'radius' parameter

$P_{10}(Y|X)$ = 0.124 at converged state, learning epoch 10

Probability distributions
For each input value X1-X5 then these get added up to a total average of 0.5+0.7+0.7+0.5+0.7 = 3.1/25 = 0.124

$P(Y|X_1)$ 'a'
1 Y_1 1|1 = 1.0
2 Y_2 1|1 = 1.0
3 Y_3 0|1 = 0.5
4 Y_4 0|1 = 0.0
5 Y_5 0|1 = 0.0

Since only 1 unit is the likely active one Out of 5 nodes

$P(Y|X_1)$ = 0.5

$P(Y|X_2)$ 'b'
Y_1 1|0 = 0.0
Y_2 1|0 = 0.5
Y_3 0|0 = 1.0
Y_4 0|0 = 1.0
Y_5 0|0 = 1.0

Since 4 units are the likely active Out of 5 nodes

$P(Y|X_2)$ = 0.7

$P(Y|X_3)$ 'c'
Y_1 1|0 = 0.0
Y_2 1|0 = 0.5
Y_3 0|0 = 1.0
Y_4 0|0 = 1.0
Y_5 0|0 = 1.0

Since 4 units are the likely active Out of 5 nodes

$P(Y|X_2)$ = 0.7

$P(Y|X_4)$ 'd'
Y_1 1|1 = 1.0
Y_2 1|1 = 1.0
Y_3 0|1 = 0.5
Y_4 0|1 = 0.0
Y_5 0|1 = 0.0

Since only 1 unit is the likely active one Out of 5 nodes

$P(Y|X_1)$ = 0.5

$P(Y|X_5)$ 'e'
Y_1 1|0 = 0.0
Y_2 1|0 = 0.5
Y_3 0|0 = 1.0
Y_4 0|0 = 1.0
Y_5 0|0 = 1.0

Since 4 units are the likely active Out of 5 nodes

$P(Y|X_2)$ = 0.7

Converged State Channel Capacity

neighborhood parameter = 0

measure	bits
H[X|Y]	1.405
H[Y|X]	1.405
I[X:Y]	0.890

```
Channel ambiguity 1.4049740389608014
Channel equivocation 1.404974038960801
BMU channel capacity 0.9999999999999999
Maximizing input source: Class:
Distribution
Alphabet:       ('1', '2', '3', '4', '5') for all
rvs
Base:           linear
Outcome Class:  str
Outcome Length: 1
RV Names:       ('X',)

x   p(x)
1   1/4
2   1/4
3   1/6
4   1/6
5   1/6
```

neighborhood parameter = 1

measure	bits
H[X|Y]	1.812
H[Y|X]	1.828
I[X:Y]	0.475

```
Channel ambiguity 1.8120559455510747
Channel equivocation 1.828097885905068
BMU channel capacity 0.9999999123774459
Maximizing input source: Class:
Distribution
Alphabet:       ('1', '2', '3', '4', '5') for
all rvs
Base:           linear
Outcome Class:  str
Outcome Length: 1
RV Names:       ('X',)

x   p(x)
1   31313663/62627327
2   1/215085921
3   1/13032264
4   1680148/6720593
5   1680148/6720593
```

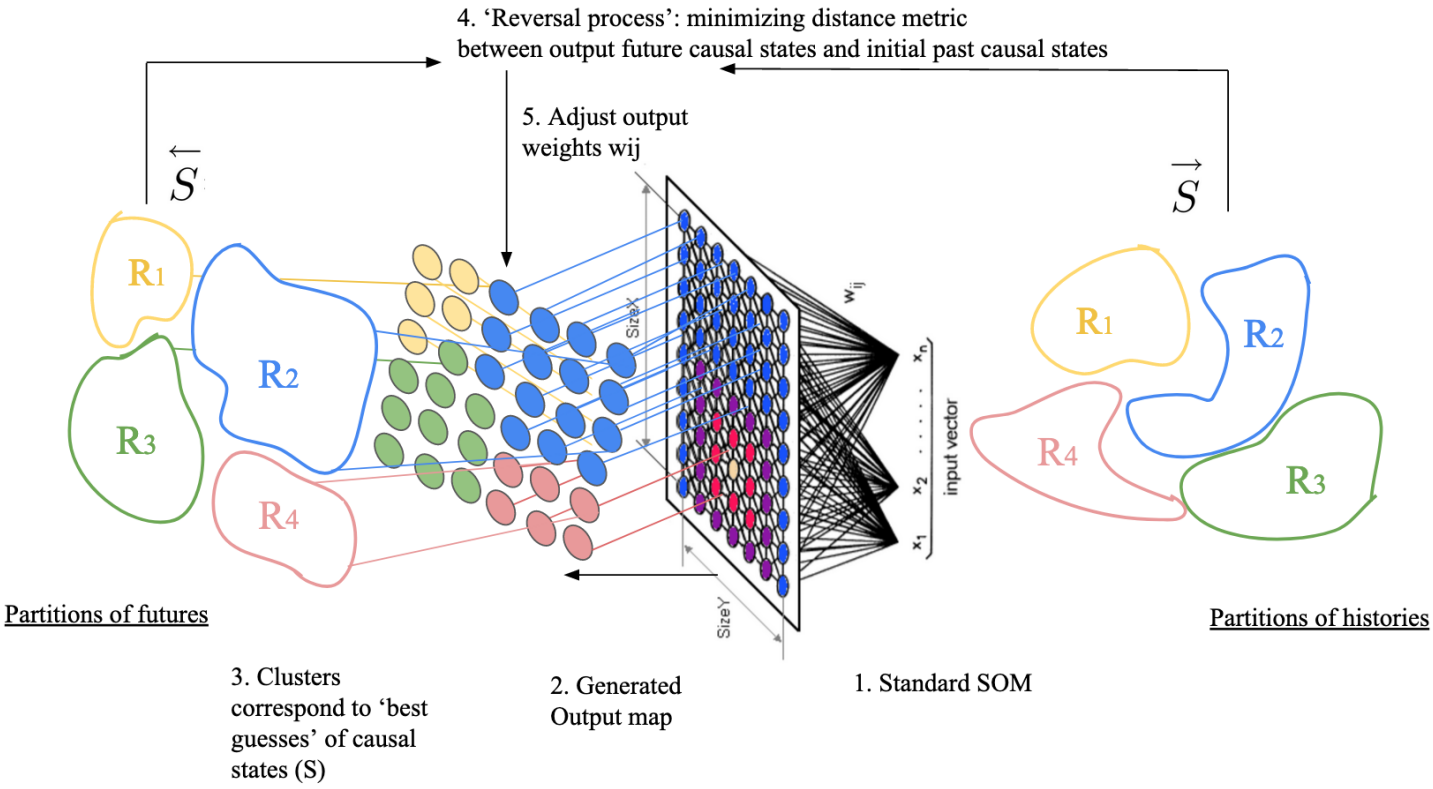
Summary Interpretation

The mutual information channel capacity gets closer to one in the converged state, so the organization of the map contributes to the mutual information ($I_0 = 0.5$, and $I_{10} = 0.89$). However, the neighborhood radius is a bit confusing, it definitely contributes to more uncertainty both in the increase of equivocation and ambiguity, this increase in uncertainty and decrease in mutual information (and channel capacity) is true of both the initial state and the converged state.

Part II: Extending the Self-Organizing Map for the reversal Process (SOeM)

We now consider a way that might automate the process of eM reconstruction, using principles inspired by the Self-Organizing Map. The self-organizing map can be thought of as a casual state map, although there are questions about how to define the boundary or the partition of each cluster. Analogously, the clustering is also a subgrouping of the inputs according to similarity, and this similarity metric can be thought of as a probability of leading to that area being “activated,” which is like a grouping of casual states. In our case we want to extend the SOM map to reconstruct eMs, which means the SOM needs a reversal process. This means the SOM needs to generate a predicted output (as if it has created a futureMorph), then it needs to

compare this output to the pastMorphs (like a reversal process). Here we do this comparison using the same SOM euclidean distance calculation between the input and output, but of the set of causal states instead of the input data to the map. Then the weights of the generated output map are adjusted until the unique best fitting eM is found. Here is a visualization of the architecture described:



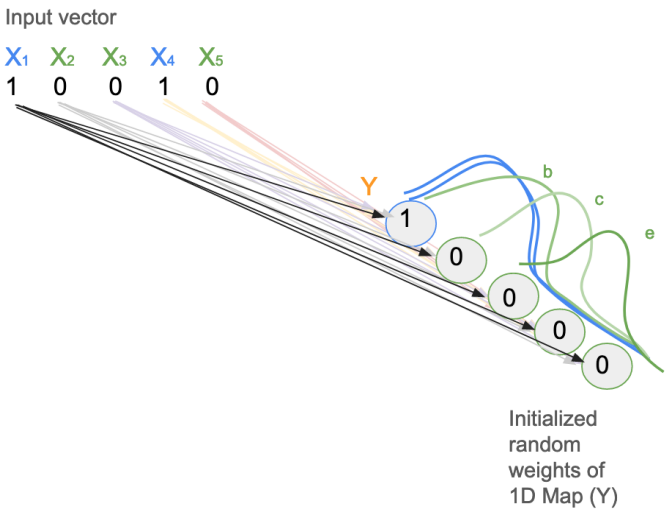
Future studies will pseudocode and implement this architecture to see if this process can generate eMs in an unsupervised way from input data.

Part III: Discussion

Technical Limitations

In part I, the measurement of channel capacity, there is still 'conceptual hygiene' to do. Particularly there are some confounds about how spatial location is involved in the calculation of

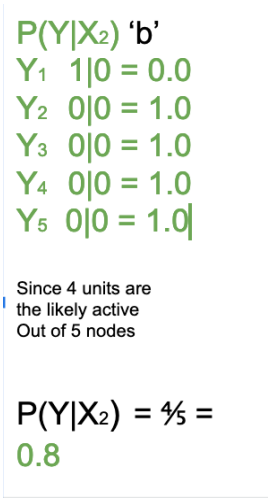
probability distributions presented here. Further general problems due to weak experience with probability theory: when looking at the initial probability curve image with the neighborhood radius parameter set to 1 (see below), the curves for each letter, drawn intuitively, imply that the probability list for those letters should be different because of their spatial location or index location, but for some reason all of these are set the same numerically:



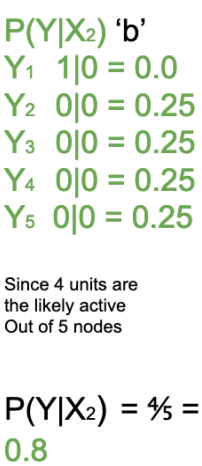
Here, the distributions c, d and e should look different as the curves imply because of their *spatial/index location*, but they are set to be the same event probabilities of activation because they are numerically the same. This is in the case where the neighborhood function is active.

Secondly, the summation of the probabilities could be different, should the total single input $X_{1\text{-}5}$ to total map Y sum to one (right), or should it be average over the total of 5 nodes (left)? Images

to explain this:



vs.



Further probability issues, I had to convert the probabilities derived in the probability distribution images to meet the requirements for dit to work, namely summing to one in the probabilities of the events, see below (left) we have the probabilities literally inputted for each alphabet event, and (right) we have a conversion/normalization of those probabilities so that the total events sum to one. Otherwise dit gives a "maximum recursion depth" warning. Hopefully despite the alterations to normalize, the underlying shape of the distribution is preserved (?).

```
#BMU as a stream of events with neighboorhood 1-step
parameter
bmuEvents = ['1a', '2a', '3a', '4a', '5a',
             '1b', '2b', '3b', '4b', '5b',
             '1c', '2c', '3c', '4c', '5c',
             '1d', '2d', '3d', '4d', '5d',
             '1e', '2e', '3e', '4e', '5e']
bmuProbs  = [1.,  0.5,  0.0,  0.0,  0.0,
             0.5,  1.,   1.,   1.,   1.,
             0.5,  1.,   1.,   1.,   1.,
             1.,  0.5,  0.0,  0.0,  0.0,
             0.5,  1,    1.,   1.,   1. ]
dist      = D(bmuEvents, bmuProbs)
dist.set_rv_names('XY')

idiagram(dist)
```

```
#BMU as a stream of events with neighboorhood 1-step
parameter
bmuEvents = ['1a', '2a', '3a', '4a', '5a',
             '1b', '2b', '3b', '4b', '5b',
             '1c', '2c', '3c', '4c', '5c',
             '1d', '2d', '3d', '4d', '5d',
             '1e', '2e', '3e', '4e', '5e']
bmuProbs  = [1./16.5,  0.5/16.5,  0.0,  0.0,  0.0,
             0.5/16.5, 1./16.5, 1./16.5, 1./16.5,1./16.5,
             0.5/16.5, 1./16.5, 1./16.5,1./16.5, 1./16.5,
             1./16.5,  0.5/16.5,  0.0,  0.0,  0.0,
             0.5/16.5, 1/16.5, 1./16.5, 1./16.5, 1./16.5]
dist      = D(bmuEvents, bmuProbs)
dist.set_rv_names('XY')

idiagram(dist)
```

Another aspect, as shown below, is that if we change how we input the alphabet our measurements may come out very different. If we reduce the input vector only to its binary alphabet we are changing a lot.

Input Vector coded as: a, b, c, d, e

```
In [12]: #BMU as a stream of events_
         # input vector as a, b, c, d, e
         # output is y1, y2, y3, y4, y5_
         bmuEvents = ['1a', '2a', '3a', '4a', '5a',
                      '1b', '2b', '3b', '4b', '5b',_
                      '1c', '2c', '3c', '4c', '5c',
                      '1d', '2d', '3d', '4d', '5d',
                      '1e', '2e', '3e', '4e', '5e']
         bmuProbs  = [1./14,  0.0,  0.0,  0.0,  0.0,_
                      0.0,  1./14, 1./14,  1./14, 1./14,
                      0.0,  1./14, 1./14, 1./14, 1./14,
                      1./14,  0.0,  0.0,  0.0,  0.0,
                      0.0,  1./14, 1./14, 1./14, 1./14 ]
         dist      = D(bmuEvents, bmuProbs)
         dist.set_rv_names('XY')

         idiagram(dist)

Out[12]: +---------+--------+
         | measure |  bits  |
         +---------+--------+
         |  H[X|Y] |  1.714 |
         |  H[Y|X] |  1.501 |
         |  I[X:Y] |  0.592 |
         +---------+--------+
```

vs

Input Vector coded as: a, b for 0 or 1

```
In [9]: #BMU as a stream of events_
        # input vector as a or b (0 or 1)
        # output is y1, y2, y3, y4, y5_
        bmuEvents = ['1a', '2a', '3a', '4a', '5a',_
                     '1b', '2b', '3b', '4b', '5b']
        bmuProbs  = [0.2,  0.0,  0.0,  0.0,  0.0,_
                     0.0,  0.2,  0.2,  0.2,  0.2]
        dist      = D(bmuEvents, bmuProbs)
        dist.set_rv_names('XY')

        idiagram(dist)

Out[9]: +---------+--------+
        | measure |  bits  |
        +---------+--------+
        |  H[X|Y] |  1.600 |
        |  H[Y|X] |  0.000 |
        |  I[X:Y] |  0.722 |
        +---------+--------+
```

This one is more true to the alphabet.

This one is more minimal?

In future studies there would be more learning epochs measured, so that there could be a nice plotted line graph of the channel capacity as the learning epochs progress through time. Here we only show the initial state and the hypothetical “converged” end state, typically reached after about 10 learning epochs. This was due to time and programming limitations. However this may be better because this allows the methodologies to be corrected before fine tuning for each epoch.

Lastly, does a probability distribution over the best matching units count as “what is being communicated” across the channel? This does make the most sense as the weights change for the best matching units, and they are the main mechanism which allows the map to organize. However, how would this capture the spatial organization as a part of what is “being communicated” by the channel as a stream of best matching units. They sort of encode their spatial location because they are associated with a node which has an index to a location and a relationship with other nodes at least when the neighborhood radius is active, but is the spatial aspect showing up at all in our measure of channel capacity? Having a more “spatially extended” information measure, or just one that accounts for geometric relationships would be interesting for the questions addressed.

Biological Plausibility

There are many reasons why the SOMs explicitly have analogies in biology, and one can read any of Kohonen's writings to see why there is plausibility. But what about the eMs? And how could this all play out on the 2D (but potentially multi-dimensional) surface of the cortex? One possible answer is that there are these areas of the cortex which are known to do mirroring of the topological representation of another area. [2] These mirror reversal representations (not to be confused with mirror neurons) are theorized to be useful to sensory integration across two or more modalities for example because for example, in certain species such as the Short-Tailed

[2] Reiner Schulz, James A. Reggia; Mirror Symmetric Topographic Maps Can Arise from Activity-Dependent Synaptic Changes. *Neural Comput* 2005; 17 (5): 1059–1083. doi: https://doi.org/10.1162/0899766053491904

Possum, the motor cortex which sends signals out, mirrors the topographic representation of somatosensory cortex, which senses signals coming in. Examples of this are also in the visual cortex of certain species. Speculatively, mirror reversal symmetries on the cortex, not only for using the same word reversal, might involve a computational process that is similar to a reversal process to achieve perhaps more localized predictive learning. If so this would be a striking example of complex predictive computations that can happen on one layer, the surface of the cortex which, although it does have lateral recurrent connectivity, is still less involved than the convolutional layered hierarchies used in Deep Learning for example.

Conceptual Significance

Understanding the information processing in self-organizing processes (part I), as well as the question of how causal inference capacities could emerge from evolutionary dynamics (sort of part II), stems from a deeper philosophical inquiry. On the one hand this would allow us to understand ourselves and how our thinking got to be as it is, as humans with a word for Turing machines and beyond, with context-free grammars and symbolic logic. Understanding how we came to be this way is a compelling question and understanding how brains, which can do causal inference, emerged out of ultimately physical laws is also contained in this question. But importantly, is the shift in our understanding of information processing that is *not* limited to humans and not limited to nervous tissues. By measuring and conceptualizing information architectures and their dynamics we might come to understand better what the limits, resources and constraints are of natural processes that can generate inferences in the world, such that we might be able to prove that an entity such a slime mold, who is single-celled, can have (or not) the information architectures capable of inferential reasoning. In other words, by understanding the architectures of information that facilitate inference we can capture more general, and likely much more *universal*, principles of learning and intelligence. This points to further potentials like biological/organic computing.

If we consider representing a set of eMs topographically using the same self-organizing process, the resultant map would be a space of possible eMs. Not to get confusing but this would be a map of maps, where the more similar eMs, similar being a topological metric and number of states etc, are clustered together. As new incoming data is processed there is more likelihood of activating a certain eM as the best fit. This would essentially be like Bayesian structural inference, but with a topological organization. The question here would be again, is there any benefit to topologically organizing the space of possible best fitting eMs? Does spatial topology contribute something, speed, efficiency of likelihoods, to inferential processes? If it does, that implies that certain kinds of spatially extended systems, with maybe specifically kinds of topological organization, are more prone or have more capacity to do inferential learning.

While this project is motivated by computational questions about the topological organization of neural tissue in the cortex of brains primarily, it could also extend to other systems because of its level of abstraction. By doing so it could expand the bounds of what can

be considered “thinking” and what is necessary to be a “thinking” entity. Perhaps one connected layer is enough, perhaps a single cell is enough complexity to “dream of electric sheep.” The question changes from: what does the topological organization do for inferential “affordances”[3]- to what is the minimal information and spatial organization needed for inference in general. How is the slime mold organizing differently than sand, than populations of neurons? If we can say something like “everything computes” and the present moment is a channel between the past and the future, what would it mean for things to “infer”, which is to form groupings of the past that represent possible futures? When a redwood tree somehow epigenetically encodes information about the weather responses it experienced over its lifetime into its seeds via the germline in such a way that the next generation of redwoods becomes more prepared for drought, is there a dynamics of predictive inference that is emergent there? In what ways could inference, or groupings of causal states (eMs), be a general kind of emergent property that plays out in physical systems not limited to nervous tissues?

[3] Gibson, James J. "The theory of affordances." *Hilldale, USA* 1.2 (1977): 67-82.