BCPNN and Sequence Learning

Ramón Martínez

March 2, 2018

1 Introduction

1.1 Sequence Learning

Section with a brief history of sequence learning in computational nueroscience. Key papers, order, structure.

The problem started with Lashley [1] Hebb reference.

The other papers

1.2 The BCPNN

Explanation of what is the BCPNN, where does it come from, and some applications.

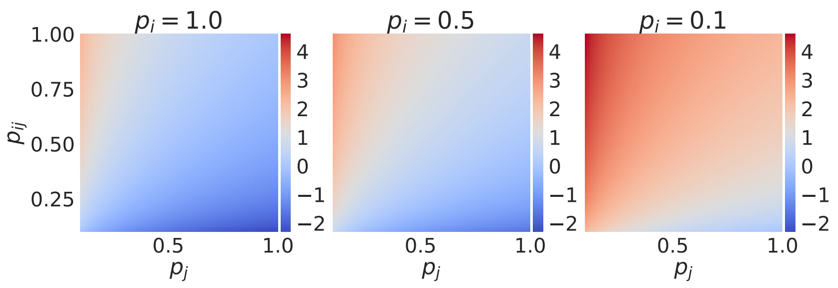


Figure 1: A schematic of how the BCPNN learning rules associates weights to different probability scenarios.

1.3 A simple phenomenological theory of sequence recall

Successful sequence recall dynamics can be described as three dynamical qualities and their interplay: fixation, erosion and biased transition.

Fixation is a general mechanism that fixes a state passing a certain point.

Examples of fixation. Attractor dynamics.

Erosion is a mechanisms that eventually suppresses the state after the system has dwell some time on it.

Examples of erosion Spike frequency adaptation [2], feedback inhibition [3].

Biased Transition, after the state has been eroded the

Examples of biased transition Biased by similiarty [3]

2 The BCPNN as a Sequence Learner

First we can show that the BCPNN can be used do incremental learning [4]. And in particular there is a spike model that can be used to learn sequences [5].

2.1 Sequence Learning with only one type of connectivity

In order to test test the capabilities of the BCPNN neural network as a sequence learning mechanism we will start with a minimal model of it. The idea here is to show the minimal conditions under which the system can successfully reproduce sequential activity and isolate how the parameters and properties of the network interact with each other. After we are equipped with this knowledge we will add more parts to the model and this in turn will provide new capabilities that we will explore later.

Using the phenomenology presented before, we will explain how the structure of the network in figure 2 and the dynamical equations of the system intertwine to achieve, fixation, erosion and biased transition altogether.

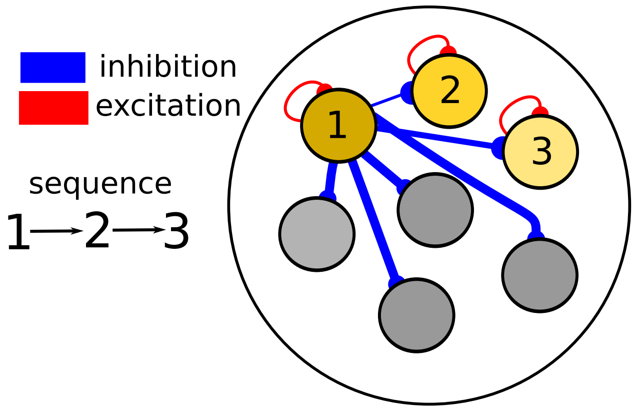


Figure 2: A simple BCPNN network with only one type of connectivity.

First the system achieves fixation by the means of the self-excitatory current depicted in red in figure 2. By itself this mechanism will fix all the patterns at the same time, that is why we need competitive selection. To solve that problem we will use a winner-takes-all mechanism [6] implemented in the form of equation 2. This equation ensures that at any point in time only the unit with the higher input current is activated.

After a particular unit is activated the adaptation current in equations 3 and 1 will be the mechanism responsible for the erosion of the pattern. Once a unit is activate for long enough and in the right parameter regime the adaptation current will surpass the self-excitatory current and the pattern will be suppressed. On this light, the time a particular unit remains activated is mostly dependent on the parameters that determine the dynamics of the adaptation and self-excitatory currents and the competitive balance between them. We will make this last relationship quantitative further down in this document.

Finally, this system would jump randomly among the states if it were not for a proper mechanism of biased transition. This is accomplished with differential inhibitory weights (illustrated in figure 2 as different widths for the blue inhibitory connections) that become more and more inhibitory the farther two units are in the sequence. This ensures that once the adaptation currents for a unit becomes big enough the next unit which is the less inhibited one wins the competition and gets prompted to activation by the winner-takes-all mechanism.

$$\tau_m \frac{ds_i}{dt} = g_{beta}\beta_i + g_w \sum_j w_{ij} o_j + g_a a_i - s_i \tag{1}$$

$$o_i = \delta_{i,argmax(s)} \tag{2}$$

$$\frac{da_i}{dt} = o_i - a_i \tag{3}$$

In order to illustrate how the dynamics of the system work together we show an example of a successful sequence recall in figure 3. In the recall process we cue the first unit of the sequence by clamping it by a short period of time ($\sim 100ms$). We then let the system evolve on its own and, given the right combination of parameters, a sequence is effectively recalled if all the units that conform the pattern are activated in the expected order.

It is important to note that in this case we utilized a tailor-made connectivity matrix to clarify the relationship between the different component of the dynamics. Further down we will show how the same effect can be effectively achieved with a matrix that emerges from a self-organized learning process.

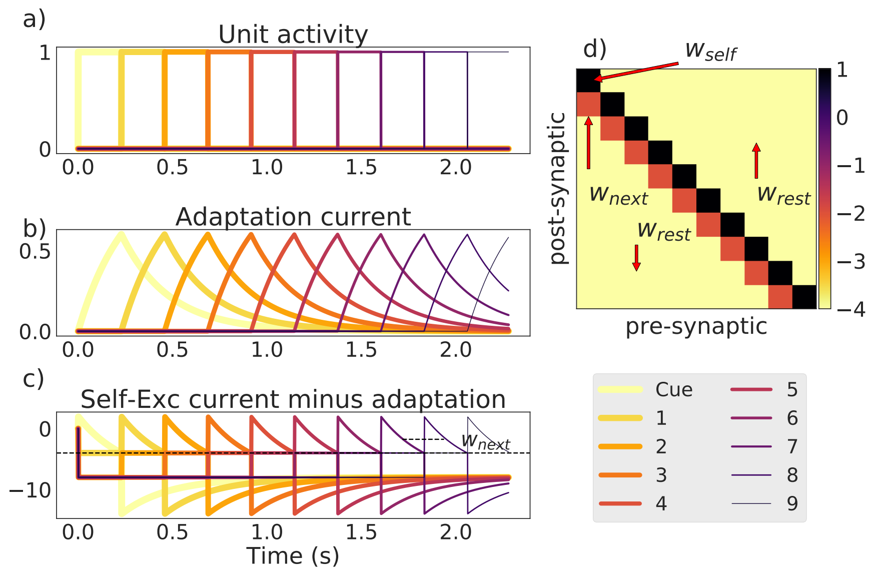


Figure 3: An instance of recall in the simple BCPNN neural network. a) Unit activity starting with the cue. b) the time course of the adaptation for each unit. c) the self-excitatory current minus the adaptation current, note that this quantity crossing the value of w_{next} (depicted here with a jagged line) marks the transition point from one unit to the next. d) The connectivity matrix where we have included pointers to the three most important quantities w_{self} for the self-excitatory weight, w_{next} for the inhibitory connection to the next element and w_{rest} for the rest of the connections.

We will now proceed to characterize the properties of the simplified version of the BCPNN. We will do this in two steps, first we will explain the recall properties of the system and then we will proceed to describe the learning rule and its dynamics. One of the most important quantities of sequence recall network is the **persistence time.** of a a state. It turns out than in this simple system by determining the recall time with regards to certain parameters we can explain both how the network works and when the recall is carried out successfully.

2.1.1 Recall properties

If we have system with only one type of connectivity, and winner-takes-all selectivity one state will be suppressed in favor of the other as soon as the support of the second state is bigger. In more detail, if we start with a system where the first unit is activated its own support will be $s_1 = gw_{self} - a(t)$ where a is the adaptation current which grows in time. This will continue as long as the first unit is activated and as a consequence the support for the unit will be decreasing. On the other hand the second unit is receiving a constant current to its support $s_2 = gw_{next}$, if this process continues by continuity there will be a point when the support of the first unit will be equal to that of the second:

$$\begin{aligned} s_1 &= s_2 \\ gw_{self} - g_a(1 - e^{\frac{t}{\tau_a}}) &= gw_{next} \end{aligned}$$

Where we have substituted the proper term for adaptation. We can solve for t in the expression above to obtain the persistent time $T_{persistent}$:

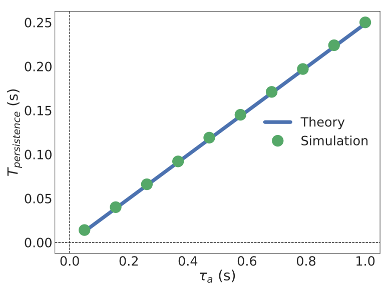
$$T_{persistence} = \tau_a \ln \left(\frac{g_a}{g_a - g_w(w_{self} - w_{next})} \right) \tag{4}$$

We notice that the equation behaves linearly with regards to τ_a and logarithmically with regards to g_a, g_w, $wself$ w_{next}. Even more importantly, certain combinations of the latter parameters will give raise to singularity points. This is important because close to this values the persistence time can be varied greatly with small variations in the respective parameters. In other words we have a wide dynamical variability in $T_{persistence}$ as in [7]. We proceed now to describe the scaling for each of the parameters in a more concise way and to compare the theory derived above with simulations.

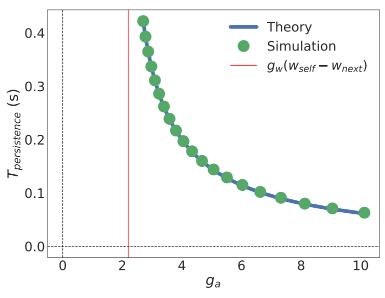
Adaptation current time constant τ_a
We explain here how the persistence time dynamics depend on τ_a the time constant of the adaptation current. A priori, the longer the adaptation time constant the longer it will take to the adaptation time current to erode the pattern. From equation 4 we can observe that the relationship is linear which is exactly what we get in figure 4 a). Note however, that the slope is not very step, that is, we do not get a lot of variation on the persistence time from variations on τ_a.

Adaptation current gain g_a The adaptation time current is actually a limiting factor in the succesfull recall of a sequence. If the adaptation current is not big enough to overcome the difference between the self-excitatory current and the current of the next element then the system will get stuck forever in the same state. We need therefore that the value of the adaptation gain g_a to be bigger the difference in weights multiplied by the weight gain $g_w(w_{self} - w_{next})$

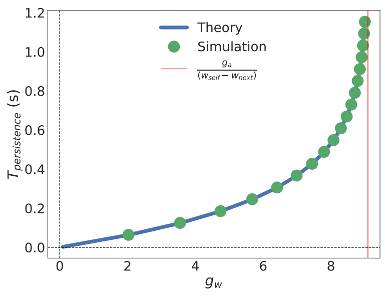
which is just the denominator of the equation 4. Once we have overcome this threshold the behavior becomes obvious, the bigger the gain of the adaptation current the faster this current will overcome the self-excitatory one, as a consequence the persistence time will be smaller. We illustrate this behavior in figure 4 b). The singularity here just reflects the fact when the adaptation is barley enough to overcome the difference in currents it takes an exponential time to actually do it.



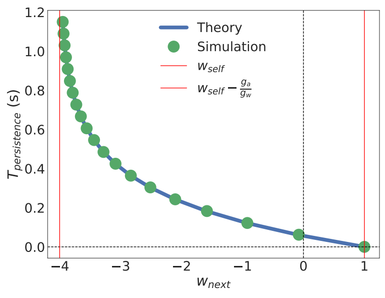
(a) τ_a



(b) g_a



(c) g_w



(d) w_{next}

Figure 4: Persistence time relationship with the parameters. a) We can appreciate that the persistence time grows linearly with τ_a, the adaptation current time constant. b) Here we depict the logarithimic dependence of $T_{persistence}$ on g_a. c) The same dependence for g_w. d) We illustrate here the effects of making the weight differential bigger (note that $w_{self} = 1$ in this plot).

Weight Gain If we look at the denominator of equation 4 we can interpret g_w as an amplifier of the weight differential ($w_{self} - w_{next}$, this in turn is a proxy for the amount of current that the adaptation current has to overcome. In the light of this is not surprising the the persistence time increases with g_w. Moreover, once the weight differential gets amplified enough the adaptation current will take an exponential amount of time to overcome this difference given rise to a singularity as well.

Next weight value In order to quantify the effects of varying w_{next} we fixed the value of w_{self} to 1. With this perspective on mind, the farther w_{next} is from 1 (the red line on the right) the biggest is the weight differential and using the same reasoning that we used for the effects of g_w above we conclude increasing persistent time and a singularity when the difference becomes big enough (red line to the left).

2.1.2 Learning Properties

Once we know the dynamics of the system given a certain matrix the natural question to consider is whether we can learn the weight matrix. As described in [4] with the help of traces we can add on-line learning capabilities to the BCPNN neural network.

$$\tau_z \frac{dz_i}{dt} = o_i - z_i \tag{5}$$

$$\tau_p \frac{dp_i}{dt} = z_i - p_i \tag{6}$$

$$\tau_p \frac{dp_{ij}}{dt} = z_i z_j - p_{ij} \tag{7}$$

$$w_{ij} = \log(\frac{p_{ij}}{p_i p_j}) \tag{8}$$

$$\beta_i = \log(p_i) \tag{9}$$

We can understand the Bayesian nature of this learning rule by focusing our attention in equation 8. The argument of the logarithm weights co-activations of the two units against the base rate activation of the units multiplied by each other. When that two units are activated at the same time the co-activation p_{ij} of the units increases, if this is bigger than the base rate of the activation of each unit multiplied ($p_i p_j$) then the weight increases, otherwise it decreases. We translate this to a problem of linking on time by using the z-traces as described in equation 5 and illustrated in equation 5. If two units are close together in time there will be co-activation of the traces (illustrated in red) within a time scale of size τ_z. Finally we use a second low-pass filter with longer time constant to preserve the learning for a longer time-scale τ_p.

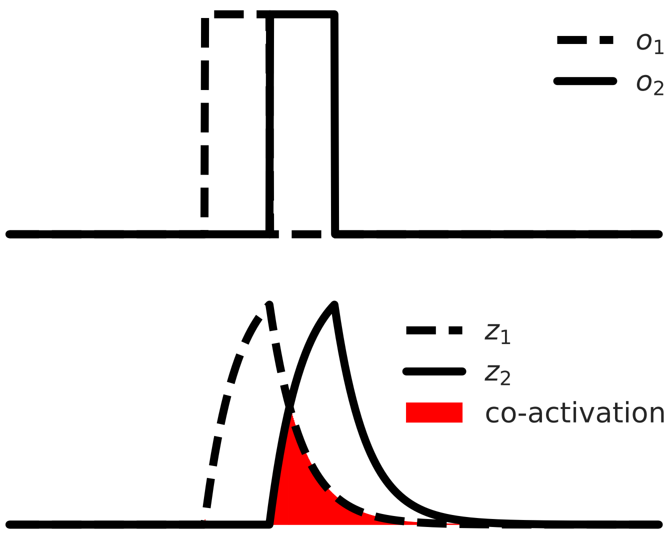


Figure 5: In red the intersection of two traces (co-activation) weighted against the base activation rate of each unit is responsible for the increase or decrease on the connectivity weight.

The training protocol:
Here we describe the training protocol. The first to note is that we train the neural network by clamping a given set of units in the order of a given sequence for a given time. More specifically a training protocol is characterized by the following quantities. First there is the **training time** for a given element of the sequence, that is, the time that element remains activated. It may well be that the an element of the sequences does not follow the other immediately, in order to account for this possibility we have the **inter pulse interval (IPS)** which is just the time elapsed between a pattern and its successor. It is also the case that a given sequence of elements is often presented more than once, we call **epoch** to a particular instance of a sequence pretension. Finally, we usually leave some time between each of the epochs during the presentation of the training protocol, this is know as the **inter sequence interval (ISI)**. We present a schematic of the training protocol in figure 6.

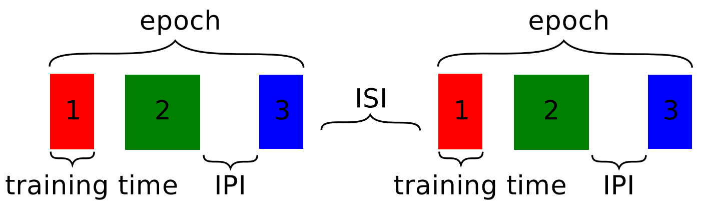


Figure 6: The training protocol. IPI stands for inter pulse interval and ISI for inter sequence interval. Explanations in the text.

In order to illustrate how this looks in practice we illustrate with a successful training example in figure 7. In this case we pass the training protocol consisting of two epochs and half a second of ISI as illustrated in b). Regarding the evolution of the weight dynamics, every time that there is a coincidence on the traces as in a) we will have a respective increase in the connectivity matrix as we see in c) marked with red shading. If the training process works properly we end up with a matrix as the one in f) which posses similar structure as the on in figure 3 d). The recall protocol consists on clamping the first pattern of the sequence for a given amount of time (T_{cue} = 100 (ms) in this case) and then letting the dynamics of the network evolve on its own. In this case we can appreciate that every element is recalled in the correct order.

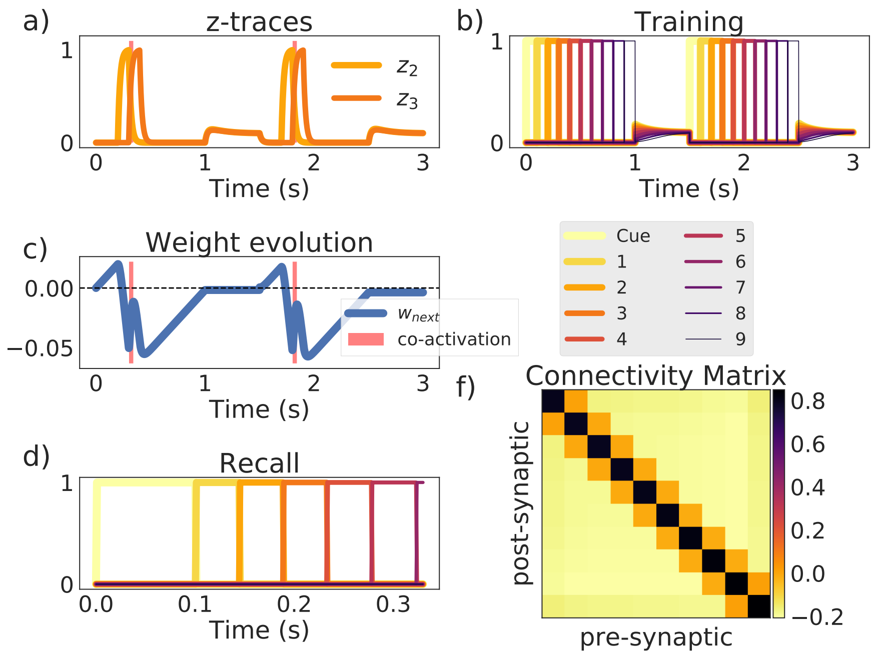


Figure 7: An example of a successful training and recall. .

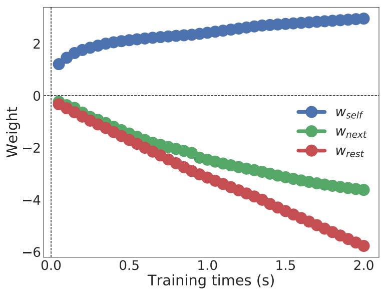
Now that we have the training protocol we will study how the different weights vary with the training protocol parameters. We ran the training protocol with a given set of parameters and we examine the values of the connectivity matrix at the end of the training protocol. We characterized the w_{self} as the self-connection of unit 2 with itself, w_{next} on the other hand is given by the weight of the connection between units 2 and 3. Finally we calculated w_{rest} as the average of the connections weights from unit two to all the other units. We studied the effects on learning of the training time, the number of epochs, the number of patterns and finally the number of units.

Training time: in figure 8 a) we show the results of learning with different training times on the connectivity matrix. Using the Bayesian interpretation of the learning rule we can explain why the w_{self} is increasing with the training time. The longer the training time the longer the unit is activated which means the co-activation of the unit with itself is bigger, which in turn leads to a bigger weight. Using the reverse of this statement is easy to see why w_{next} and w_{rest} are decreasing. The BCPNN learning rules weights the co-activation of the units against their individual activation. As the training time grows for different units the latter becomes bigger and bigger and therefore the decreasing effect on weights that we observe on the graph.

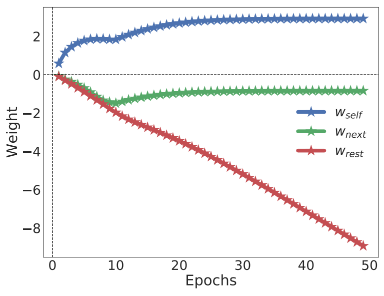
Epochs: in figure 8 b) we show the results that training with different number of epochs has on the connectivity matrix. This graph is better understood in terms of the dynamics of learning reaching to their natural equilibrium. The co-activation to independent activation ratio is dynamical process that follows the equations, the longer the dynamics run the closer this dynamic approaches its steady-state behavior. In the case of w_{self} and w_{next} there are definitive ratios of activation given by the dynamic and the system converges to it. The w_{rest} quantity however, does not. This reflects the fact that the units never activate together and the learning rule keeps accumulating evidence of it.

Number of patterns: in figure 8 c) we depict the outcome that having a different number of patterns in the sequence has on the connectivity matrix. In terms of the Bayesian nature of the learning rule this graphs can be explained in the following way. When we increase the number of patterns we are making the overall probability space bigger (number of possible ways in which the network could possible be). This makes the any two units co-activation every more meaningful, this is the reason why w_{self} and w_{next} increase. On the other hand, the network has more and more time to accumulate evidence of the lack of co-activation for the rest of the units and therefore w_{rest} decreases.

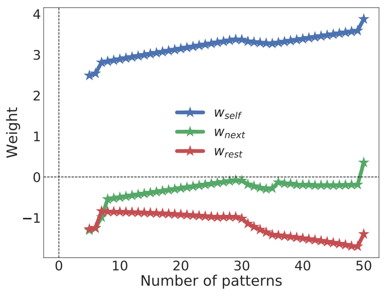
Number of minicolumns in figure 8 d) we illustrate the effects of the number of units on the connectivity matrix. The reasoning is analogue to the one in the number of pattern, but the effect is more pronounced because we are making the network bigger overall.



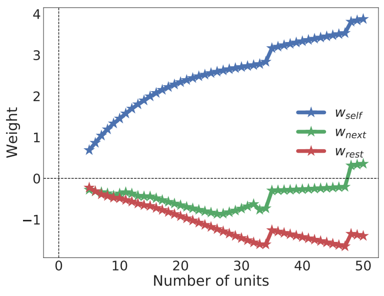
(a) Training time



(b) Epochs



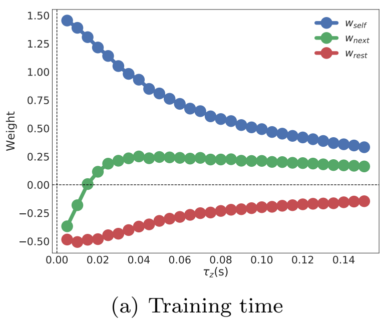
(c) Number of patterns



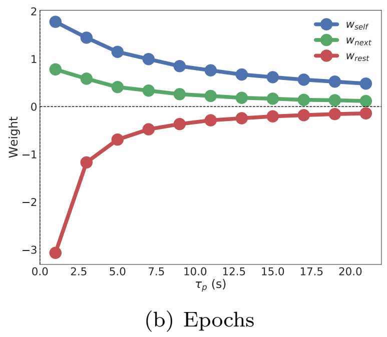
(d) Number of units

Figure 8: Dynamics of weight learning. a) effects of the training time on learning. b) effects of the number of epochs on training. c) effects of the number of patterns on learning. d) effects of the number of units on learning. See text for explanation.

Effects of learning on the persistent time: in terms of the dynamics of the persistence time we can sketch out how the different training regimes will affect the behavior of the former quantity. The effect of an increased training time increases the distance between w_{self} and w_{next}, this will increase the time that it takes the



(a) Training time



(b) Epochs

Figure 9: Dynamics of weight learning. a) effects of the training time on learning. b) effects of the number of epochs on training. c) effects of the number of patterns on learning. d) effects of the number of units on learning. See text for explanation.

3 The problem Of Complex Sequences

Here it is one paper [8]

References

[1] K. Lashley. The problem of serial order in behavior. In *Cerebral mechanisms in behavior*, pages 112–136. 1951.

[2] James P Roach, Leonard M Sander, and Michal R Zochowski. Memory recall and spike-frequency adaptation. *Physical Review E*, 93(5):052307, 2016.

[3] Stefano Recanatesi, Mikhail Katkov, and Misha Tsodyks. Memory states and transitions between them in attractor neural networks. *Neural computation*, 29(10):2684–2711, 2017.

[4] A Sandberg, A. Lansner, K. Petersson, and O. Ekeberg. A bayesian attractor network with incremental learning. *Network: Computation in neural systems*, 13(2):179–194, 2002.

[5] P. Tully, H. Lindén, M. Hennig, and A Lansner. Spike-based bayesian-hebbian learning of temporal sequences. *PLoS computational biology*, 12(5):e1004954, 2016.

[6] Alan L Yuille and Davi Geiger. Winner-take-all mechanisms, the handbook of brain theory and neural networks, 1998.

[7] J. M Murray et al. Learning multiple variable-speed sequences in striatum via cortical tutoring. *eLife*, 6:e26084, 2017.

[8] I Guyon, L Personnaz, JP Nadal, and G Dreyfus. Storage and retrieval of complex sequences in neural networks. *Physical Review A*, 38(12):6365, 1988.