4/12/17

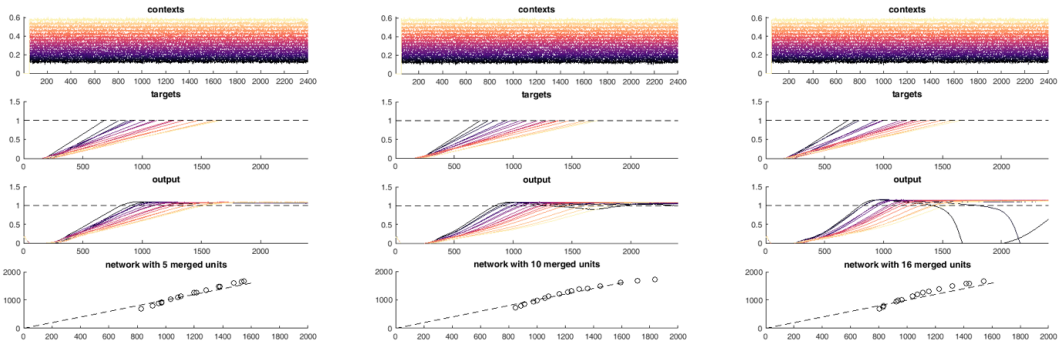
Goal: Train RNN to learn temporal integration task

4/17/17

Goal: Retrain network to decrease the unit number

4/22/17

Goal: Try collapsing the network to decrease unit number without retraining



4/28/17

Goal: How does the richness of the solution depend on the number of units?

5/1/17

Goal: Review properties of LIF neurons and write some code to run this model forward

---

5/6/17

Goal: Reproduce results from Fiete et al., in python

Spike-Time-Dependent Plasticity and Heterosynaptic Competition ...

Their model:
HVC_{RA} neurons have recurrent synapses that can be modified. HVC_I consists of a single inhibitory unit which connects to all units in the network. Weights are initialized random and small and inputs to the excitatory cells are likewise random and uncorrelated (white). The main process is that excitatory cell undergo asymmetric STDP (similar to the kernel used in the RL paper with Bence) and there is some level of heterosynaptic competition.

The kernel in Fiete et al.,

$$K(t) = e^{-t/\tau_{stdp}} \tag{1}$$

Where they use different initial conditions aka $K(0)$, depending on whether they are using a spiking model or a binary neuron model.

The kernel in Tesileanu et al.,

$$\frac{dW_{ij}}{dt} = \eta \hat{c_i}(t)(g_j(t) - \theta) \tag{2}$$

$$\hat{c_i}(t) = \int_0^t dt' c_i(t)[\frac{\alpha}{\tau_1} e^{-(t-t')/\tau_1} - \frac{\beta}{\tau_2} e^{-(t-t')/\tau_2}] \tag{3}$$

This is the sum of two exponenetials which can be parametrically varied. I think this would be a fun thing to do with the Fiete model to see how robust it is to different kinds of STDP rules (note: I should look at what the different evidence is for STDP between HVC RA neurons).

The full weight update rule used in Fiete et al.,

$$W_{ij}(t) = W_{ij}(t-1) + \eta \Delta_{ij}^{STDP}(t) - \epsilon\eta\theta_{i*}(t) - \epsilon\eta\theta_{*j}(t) \tag{4}$$

where ΔW_{ij}^{STDP} is given by:

$$(\frac{W_{ij}}{W_{max}} + 0.001) * [x_i(t)K(0)x_j(t) + \sum_{\tau=1}^{T} x_i(t)K(\tau)x_j(t-\tau) - x_i(t-\tau)K(\tau)x_j(t)] \tag{5}$$

This is will give the behavior that if x_j fired in some window before x_j fired, we will in increase the weight. Otherwise if this is reverse we will decrease the weight.

Finished reading and running the paper code, but I haven't got mine to work entirely yet.

5/7/17

Goal: Understanding simple recurrence

Make toy two neuron networks and analyze the stability of the network dynamics by diagonalizing the weight matrix

Our dynamics will be described by this differential equation:

$$\tau\dot{\vec{x}} = -\vec{x} + W\vec{x} + \vec{h} \tag{6}$$

This code is to reproduce what michale showed in class to get the dynamics of a simple two neuron recurrent network with the the following symmetric weight matrix:

0 0.8
0.8 0

See *Toy Network.ipynb*

5/8/17

Goal: Make a ring attractor in python, test stability and get the bump to move