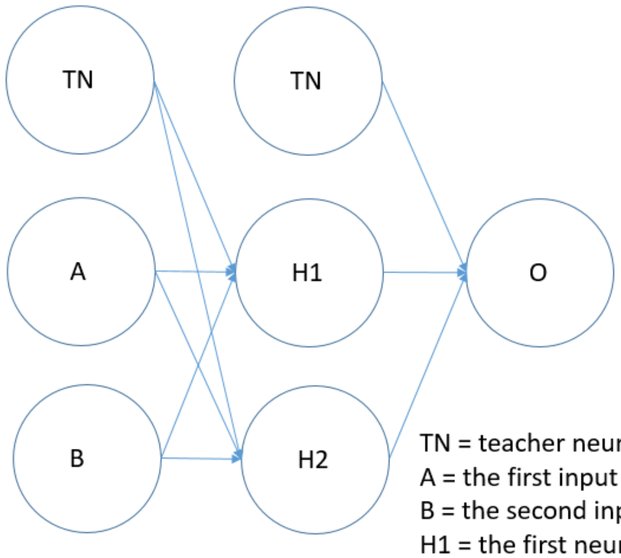
Jessie George
Net ID: jg1114
Brain Inspired Computing
Homework 2

Part 1

1. Temporal encoding of a neuron records the timing of the spikes [in milliseconds]. Let's say a presynaptic neuron fires at time t_j^f and the postsynaptic neuron fires at time t_i^f then the resulting change in the synaptic strength Δw_{ij} is a function of the difference t_j^f - t_i^f. If the presynaptic neuron fires shortly before the postsynaptic neuron, the strength of the synapse increases. If the reverse occurs, then the strength of the synapse decreases.

2. Firing rate encoding of a neuron records the number of spikes in a trial divided by the time duration of that trial when current is injected or received.

3. I chose to use LIF neurons. (The LIF neuron code is in LIF.py). Diagram for XOR gate:



TN = teacher neuron
A = the first input neuron
B = the second input neuron
H1 = the first neuron in the hidden layer which is a NAND gate
H2 = the second neuron in the hidden layer which is an OR gate
O = the output neuron which is an AND gate

XOR is the same as the AND of the result of NAND and the result of OR

4. The code is attached in XORHebb.py. The output is attached in XORHebbOutput.txt.
I used the Hebbian learning equation from lecture 5 slide 43:
dw_ij/dt = a_2_corr * v_j_pre * (v_i_post - theta)
The output shows the following:
first input neuron spike timings
second input neuron spike timings
(Note: for the spike timings array, at the nth subscript, it is 1 if neuron spiked at nth second, else it is 0)
the XOR output
the learned output

5. The code is attached in XORstdp.py. The output is attached in XORstdpOutput.txt.
   In order to increase/decrease the weights, I used the difference in spike timing equation from lecture 6 slide 42:
   $t_j^f - t_i^f$
   The output shows the following:
   first input neuron spike timings
   second input neuron spike timings
   (Note: for the spike timings array, at the nth subscript, it is 1 if neuron spiked at nth second, else it is 0)
   the XOR output
   the learned output
   For each synapse it shows the spike train of the presynaptic neuron, the spike train of the postsynaptic neuron, and the weights for the synapse.

Part 2
The code is attached in LDN.py and OnOffCell.py
The output is attached in LDNoutput.txt
(LDN meaning Line Detector Neuron)
A line detector has a sequence of on cells sandwiched between 2 sequences of off cells.
See the example below where x = off cell and o = on cell:
x x x
o o o
x x x
The ideal line aligns with the LDN's receptive field perfectly. A non-ideal line will only be partially inside the LDN's receptive field.
I made a horizontal line detector for a matrix of size 5x7 which is 9 cells.
Each sub-grid is size 3x3 which represents a single cell (which could be an on cell or an off cell, or neither).
The boolean list is set to true if that single cell is an on cell (as determined by the neuron in OnOffCell.py)
If the line coincides with the receptive field of the Line Detector Neuron, i.e the 3 horizontal on cells fire, then the output says that the matrix contains a horizontal line, else it doesn't.
I tested it with lines of angles in 20 degree increments as shown in the output.