Report of Assignment 2

Haoyang Zhang (hz333), Qi Wu (qw142), Shengjie Li (sl1560)

May 5, 2020

Problem1: XOR

1. `1` : input neuron will fire with a delay of 6ms.
   `0` : input neuron will fire with a delay of 0ms.

2. `1` : input neuron will fire with a higher rate, like `200 Hz`.
   `0` : input neuron will fire with a lower rate, like `100 Hz`.

3. The 3-layer SNN (code available in `./code/xorBCM.py` and `./code/stdp.py`) we build including 2 Poisson neurons at input layer, several LIF neurons at hidden layer and 1 LIF neuron at output layer. The exact network topology is introduced in our answers to the next two questions.

4. In Hebbian learning part (code available in `./code/xorBCM.py`), the coding method of input and output is rate coding. More specifically, we encode input `1` with `200 Hz`, and `0` with `100 Hz`. For the output decoding, we use `150 Hz` to represent `1`, and `40 Hz` to represent `0`. The threshold between `1` and `0` is set as `95 Hz`. The network topology is $(2, 2, 1)$. (See Fig. 1.) We use BCM rule to train the network. To train the network, we add 3 teaching input currents totally, 2 for the hidden layer and 1 for output layer. The teaching input currents directly inject into the "student" neuron to force it firing or not. We constrain the hidden neuron to represent $h_1 = \neg x \wedge y, h_2 = x \wedge \neg y$, and therefore the output neuron acts as $z = h_1 \vee h_2$. (See Table. 1) For ease of training, we further trained this network layer by layer, and constrained the weights being symmetric in terms of x and y.

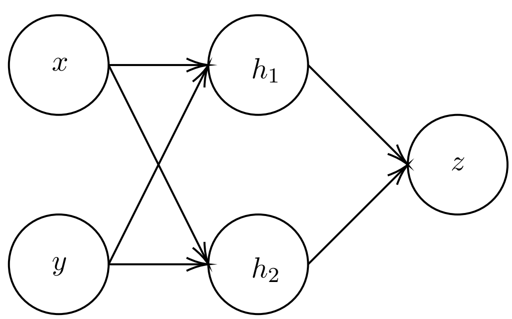
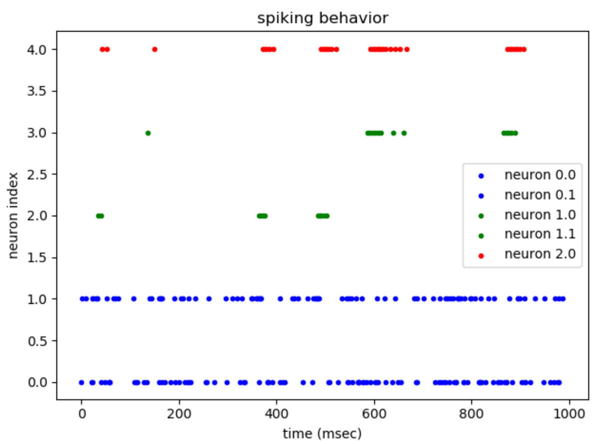


Figure 1: the topology of xorBCM network

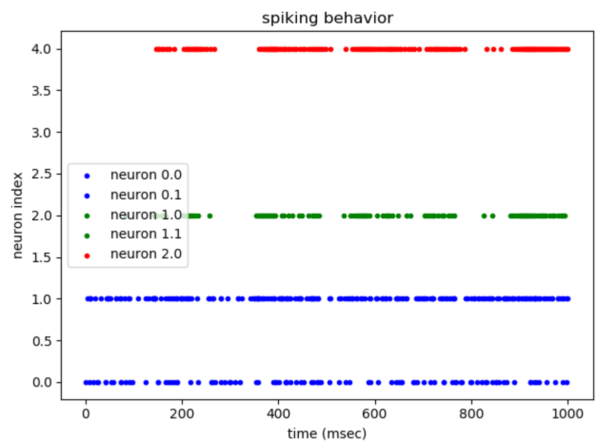
The result is shown in Fig. 2 and 3. The output neuron fires at averaged about `50 Hz` with input `(0, 0)`, `160 Hz` with `(0, 1)`, `150 Hz` with `(1, 0)`, and `40 Hz` with `(1, 1)`. Namely, the output neuron performs the $\oplus$ operation of the inputs.

x	y	h_1	h_2	z
0	0	0	0	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

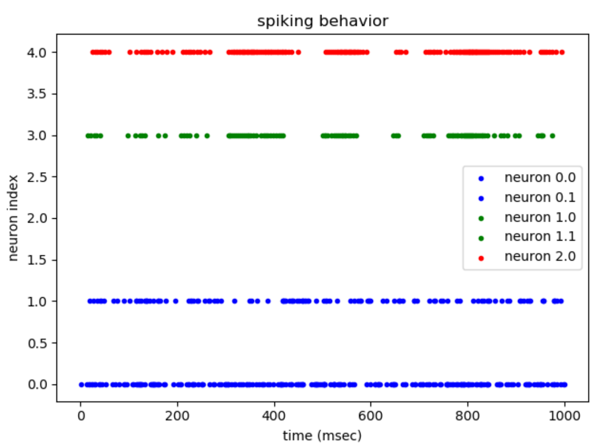
Table 1: the truth table of xorBCM network



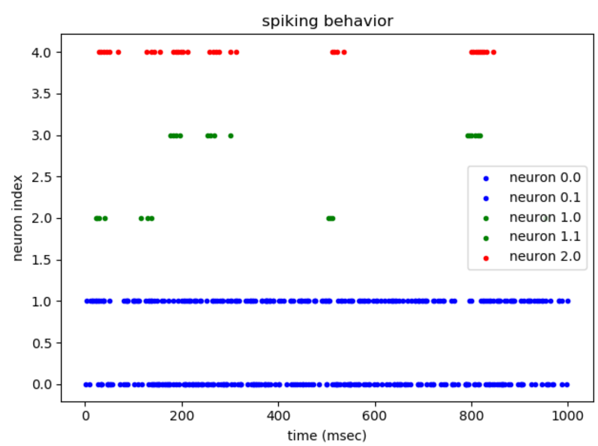
(a) input `(0, 0)`



(b) input `(0, 1)`



(c) input `(1, 0)`



(d) input `(1, 1)`

Figure 2: spiking behavior of xorBCM network

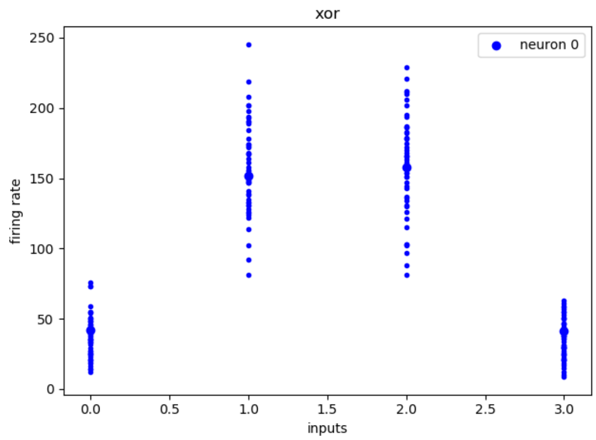


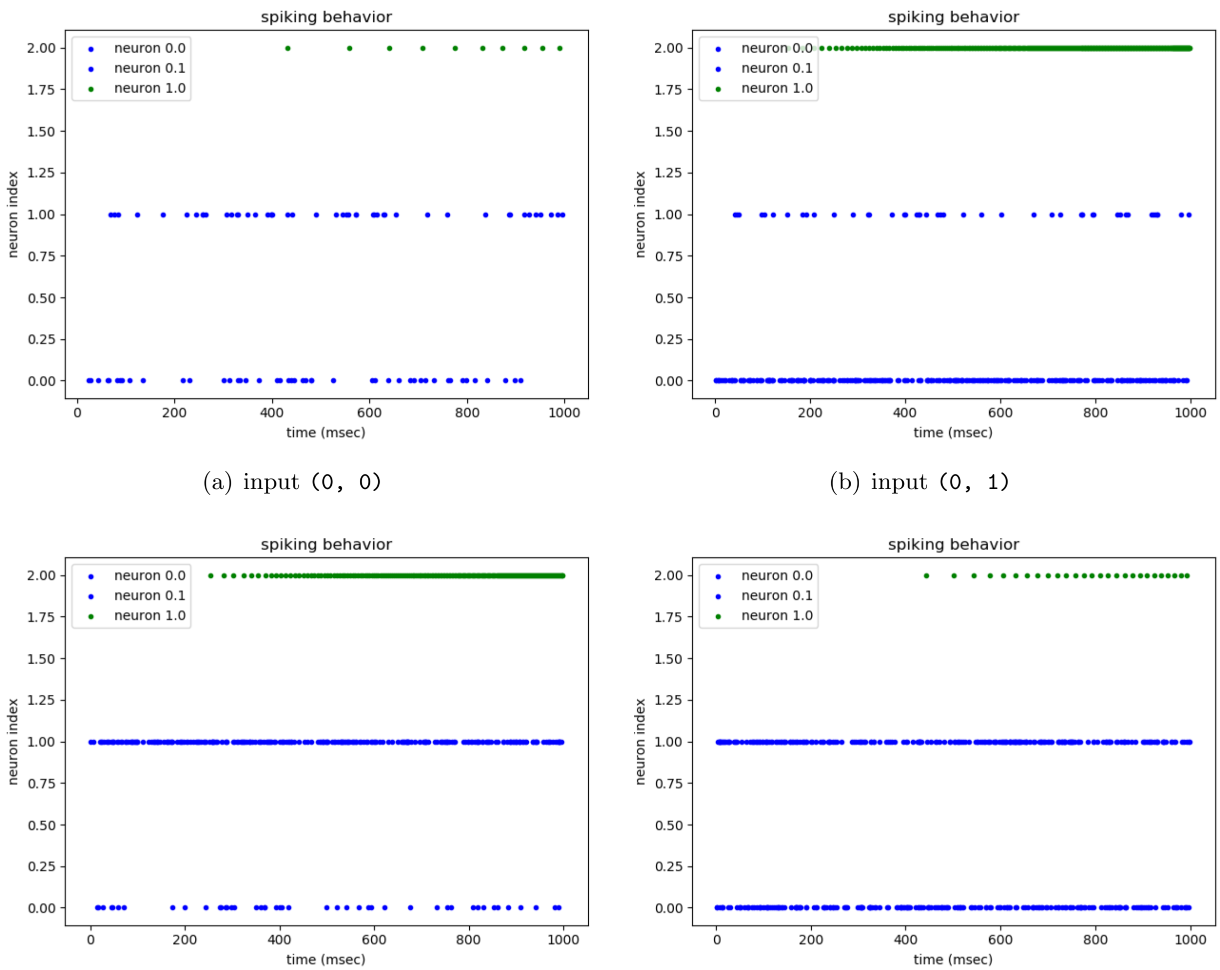
Figure 3: averaged spiking rate of xor neuron (trained by BCM) of 50 iterations, where inputs 0.0, 1.0, 2.0, and 3.0 represent input `(1, 1)`, `(1, 0)`, `(0, 1)`, and `(0, 0)`, respectively

5. In STDP learning part (code available in `./code/stdp.py`), we also use rate coding to code the input and output. For the input neuron, we encode `1` with `125 Hz`, and `0` with `25 Hz`. For output neuron, `1` fall in range `100-150Hz`, and `0` fall in `0-30HZ`. The network topology is $(2, 10, 1)$. Also, we add a teaching input as supervised data to train the network. For teaching input, we use `250 Hz` to represent `1` and `50 Hz` to represent `0`. In training, we first run the network with 1000 ms and get the spike train of each neuron. Next, we update the weight of synapse between the input layer and the hidden layer with unsupervised STDP (Morrison et al., 2008). Then, we use the teaching input to update the synapse between the hidden layer and the output layer with supervised STDP, ReSuMe (Ponulak and Kasiundefinedski, 2010). The update rule is as follows:

$$\Delta w(t) = [S_d(t) - S_o(t)][a_d + \int_0^{\infty} a^{post} S_h(t-s)\mathrm{d}s]$$

   Where S_d, S_o, and S_h are the spike train of the teaching input, the output neuron and the hidden neuron. We set $a_d = 0$ and $a^{post} = 0.005$.

   The result is shown in Fig. 4.



(a) input `(0, 0)`

(b) input `(0, 1)`

(c) input `(1, 0)`

(d) input `(1, 1)`

Figure 4: spiking behavior of STDP network (neurons 0.0 and 0.1 are Poisson input neurons and neoron 1.0 is the output neuron)

Problem2: Line detector

We construct the "on-off" cell with two parts: the convolutional part and the Poisson part. The convolutional part takes a portion of the whole inputs and convolutes it with a kernel (See Table. 2.) to perform the receptive field of the "on-off" cell. Then, Poisson part takes the output of the convolutional part as inputs, and emits spikes with the rate proportional to inputs (in range `10 Hz - 200 Hz`). (See Fig. 5.) To decode the outputs, we still use the rate coding scheme. Specifically, we set the threshold as `150 Hz`.

-1	-2	-1
-2	12	-2
-1	-2	-1

Table 2: convolutional kernel of "on-off" cells

For ease of denotation, call the "on-off" cells that are excited if on-center, off-surround On Cell,

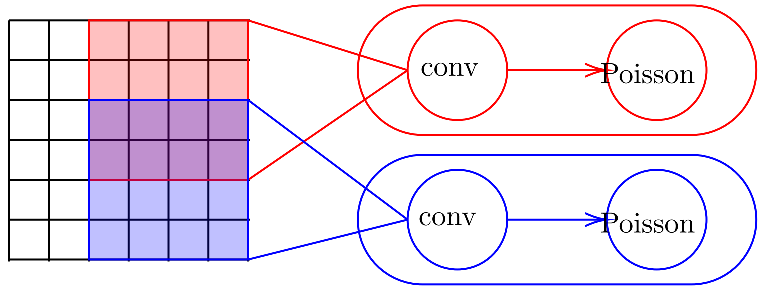


Figure 5: two “on-off” cells

and the other type Off Cell.

We consider building an line detector to detect an horizontal line. Namely, the detector neuron will be excited if a row of On Cells are excited, which indicates existing at least a line, and the two rows of Off Cells, next to the given row, are also excited, which indicates existing no more than a line. Therefore, the topology of this network is sparse. (See Fig. 6.) More specifically, we set the weight of On Cells to detector cells as 1, and Off Cells to detector cells as 0.5.

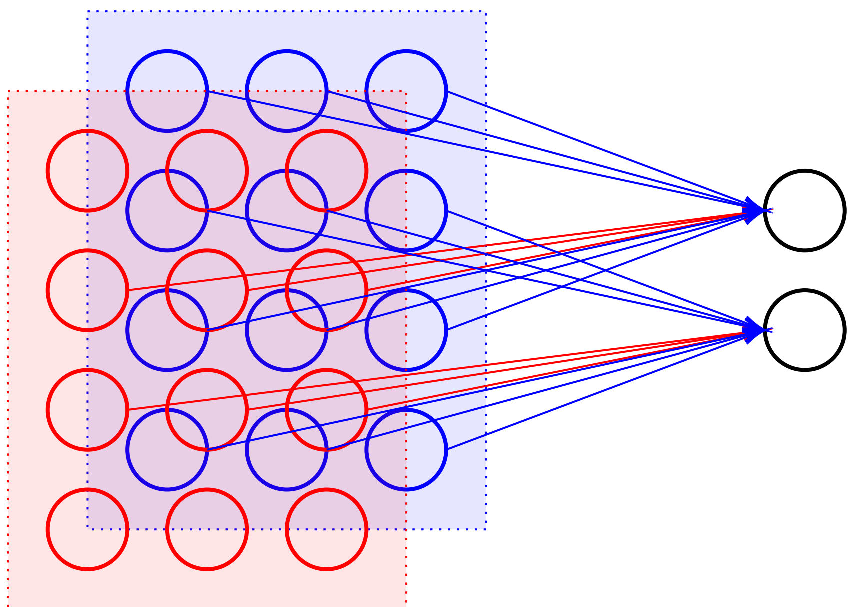
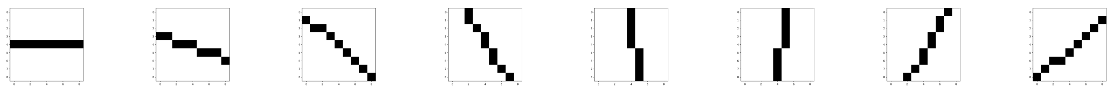


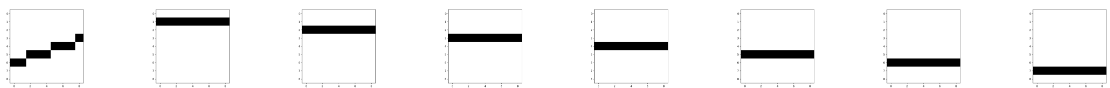
Figure 6: the topology of line detector network

Then, we tested this network with different angles (See Fig. 7(a)-7(i).) and positions (See Fig. 7(j)-7(p)). Note that because of a line is centrosymmetric, $-80^\circ = 100^\circ$, we only need these 9 inputs (Fig. 7(a)-7(i)) to cover all angles with 20 degree increments. The result is shown as Fig. 8, and detailed spiking behavior plots can be found in `./docs/plots/`.

Note that neuron 4 is potentially excited with the input as Fig. 7(b) and Fig. 7(i), because these two inputs cantains a line segment at row 5. But the firing rate is still under `150 Hz`, which does not violate the output decoding threshold.

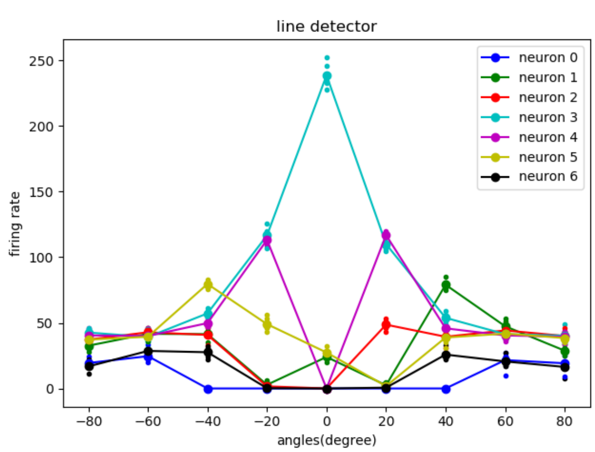


(a) (b) (c) (d) (e) (f) (g) (h)

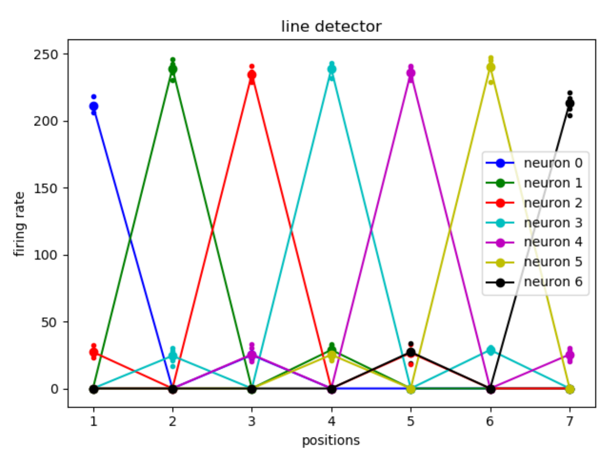


(i) (j) (k) (l) (m) (n) (o) (p)

Figure 7: inputs of line detector networks



(a) spiking rate for different line positions



(b) spiking rate for different line positions

Figure 8: averaged spiking rate of line detector neurons of 5 iterations

References

Abigail Morrison, Markus Diesmann, and Wulfram Gerstner. 2008. Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, 98(6):459–478.

Filip Ponulak and Andrzej Kasiundefinedski. 2010. Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting. *Neural Comput.*, 22(2):467–510.