

Distributed Representations & Learning

Problem Set 10-17

October 17, 2003

Due: Monday October 27

Where's the point?

In this problem, we consider a simple connectionist network model of visuo-motor coordination. The network's input is a position in space, and the output is 'muscle stimulation' to 'point' to that position.

Question 1: Setting up the model

1a Inputs

In this simple model, only one dimension of space will be considered. Along this dimension are four positions which we'll call (a), (b), (c), and (d), from left to right. There are four input units which we'll call ①, ②, ③, and ④; their activation values are $i_1, i_2, i_3,$ and i_4 . The receptive fields of these four units are shown in Figure 1. [The receptive field of ① includes all four positions; the receptive field of ② includes (a), (b), and (c); the receptive field of ③ includes (b), (c), and (d); the receptive field of ④ includes (c) and (d).]

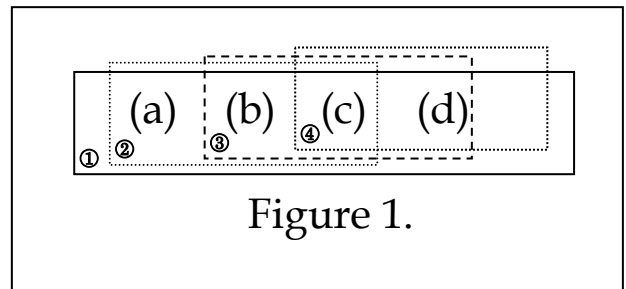


Figure 1.

In this problem, we will assume that an input unit is either fully active or fully inactive, with activation value either 1 or 0. The representation of a given position (p) is the pattern of activation in which each input unit with (p) in its receptive field is active (activation value = 1), and all other input units are inactive (activation value = 0). Give the activation vectors representing each position, listing the activation values in the obvious order (i_1, i_2, i_3, i_4).

Input vectors $i^{(p)}$:

<u>Position</u>	<u>Activation vector</u>
(a)	_____
(b)	_____
(c)	_____
(d)	_____

1b Outputs

The output of the network is 'muscle stimulation' modeled as follows. There are four output units, each of which is either fully active (1) or fully inactive (0); we'll call their activation values $o_1, o_2, o_3,$ and o_4 . These units send their activation to a muscle, and the more activation the muscle receives, the more it contracts and the further to the left a hypothetical 'arm' 'points'. To 'point' to position (a) at the far left, maximal stimulation is required: all four output units must be active. To point to (b), three must be active; we'll assume it's the first three; the fourth must be inactive. To point to (c), only the first two units must be active; to point to (d), only the first must be active. Give the output vectors for pointing to each position, listing activation values in the order (o_1, o_2, o_3, o_4).

1c Hyperacuity

Suppose we can get a network to perform this task correctly. Then how does the acuity of the network – the smallest difference in input position that it can correctly respond to – compare to the size of the receptive fields of individual input units? How could this be possible?

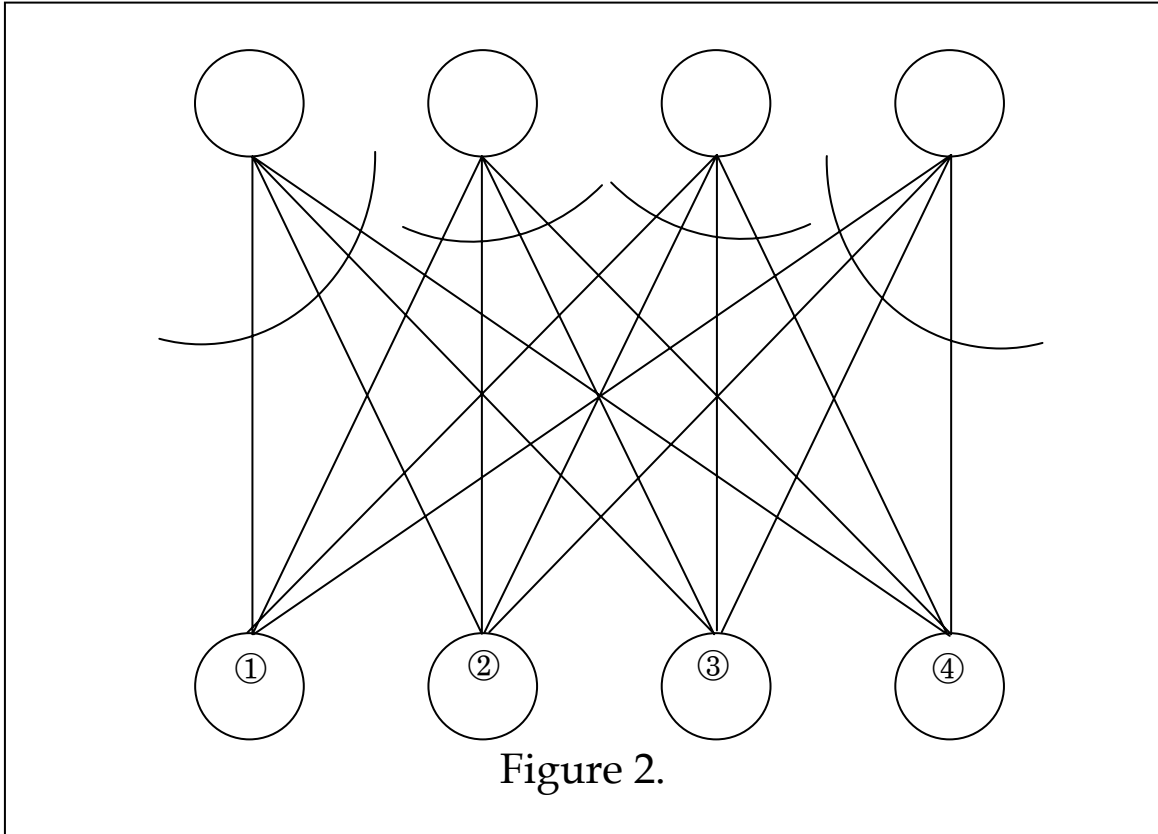
Correct output vectors $o^{(p)}$:

<u>Position</u>	<u>Activation vector</u>
(a)	_____
(b)	_____
(c)	_____
(d)	_____

Question 2: Hebbian learning

2a Computing the Hebbian weights

Now compute the weights learned by Hebbian learning, assuming that each input/output vector pair (one for each position) is presented once, with a learning rate $\epsilon = 0.5$, starting from zero initial weights. Show these weights in Figure 2 (on the arcs). Show your work on an attached sheet, labeled '2a'.



2b Testing the Hebbian weights

For each of the four positions (p), clamp the input vector representing (p) on the input units of your network in Figure 2, and compute the output vector using the Hebbian weights in Figure 2. Assume the output units are linear. Is the network computing the correct outputs? Why or why not? As in 2a, and for the rest of the exam, show your work on an attached sheet with the problem part clearly labeled.

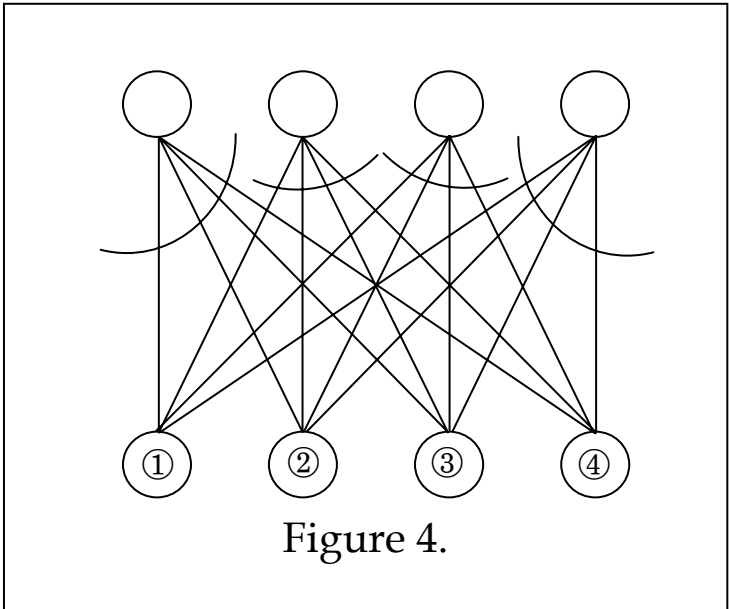
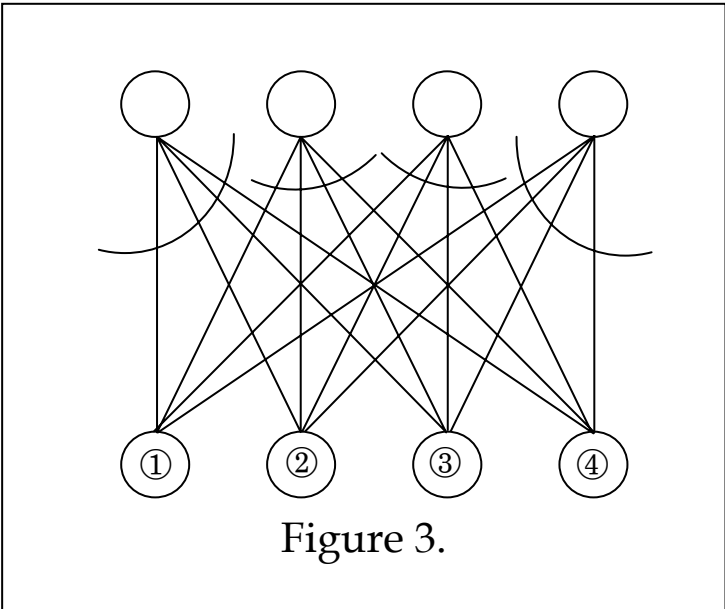
Hebbian-computed output vectors $\mathbf{o}^{-(p)}$:	
<u>Position</u>	<u>Activation vector</u>
(a)	_____
(b)	_____
(c)	_____
(d)	_____

Question 3: δ -Rule Learning

3(a)(b) Computing the δ -rule weights

Now let's train the network with δ -rule learning instead of Hebbian learning. (We will work out only the very beginning of the process.) Again, we start with zero weights and assume a learning rate of $\epsilon = 0.5$. Present the input/output vectors for position (a), and compute the resulting δ -rule weights; put the results in Figure 3. Then present the input/output vectors for position (b), and put the resulting δ -rule weights in Figure 4. Complete the following table along the way.

δ-rule computation				
<u>Position</u>	Input vector	Correct output vector \mathbf{o}^+	Computed output vector \mathbf{o}^-	δ vector
(a)	_____	_____	_____	_____
(b)	_____	_____	_____	_____



Question 4 Extra credit: Final δ -rule weights

Already in presenting the second input/output pair (b) in 3(b) above, certain positive weights began to decline, and certain zero weights went negative. In the final δ -rule weights we now compute, we'll see that certain of the former weights eventually go to zero and the latter weights eventually become as strongly negative as others are positive. We will compute the final δ -rule weights using the following 'perpendicular vectors':

$$\mathbf{w}^{\perp(a)} = (1 \ 0 \ -1 \ 0) \quad \mathbf{w}^{\perp(b)} = (0 \ 0 \ 1 \ -1) \quad \mathbf{w}^{\perp(c)} = (-1 \ 1 \ 0 \ 1) \quad \mathbf{w}^{\perp(d)} = (1 \ -1 \ 0 \ 0)$$

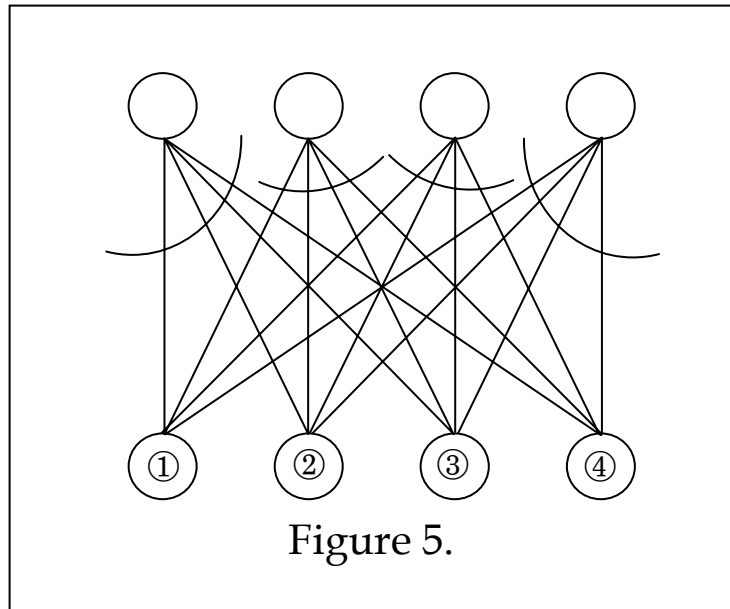
4a The right \angle stuff

Verify that these vectors satisfy the requirements defining them (from lecture):

$$\mathbf{w}^{\perp(p)} \cdot \mathbf{i}^{(p')} = 0 \quad \text{for } p' \neq p, \text{ and } \mathbf{w}^{\perp(p)} \cdot \mathbf{i}^{(p)} = 1$$

4b Skipping to the last page ...

Compute the final δ -rule weights, and write them in Figure 5.



4c Happy ending (of course)

Verify that the final δ -rule weights compute the correct answer for each of the four positions (a)–(d).

Final δ -rule network performance				
Position	Input vector	Correct output vector \mathbf{o}^+	Computed output vector \mathbf{o}^-	δ vector
(a)	_____	_____	_____	_____
(b)	_____	_____	_____	_____
(c)	_____	_____	_____	_____
(d)	_____	_____	_____	_____