Kohonen's Self Organizing Map: Reduction of Dimension

Akira Imada

1 Map from 2D to 2D

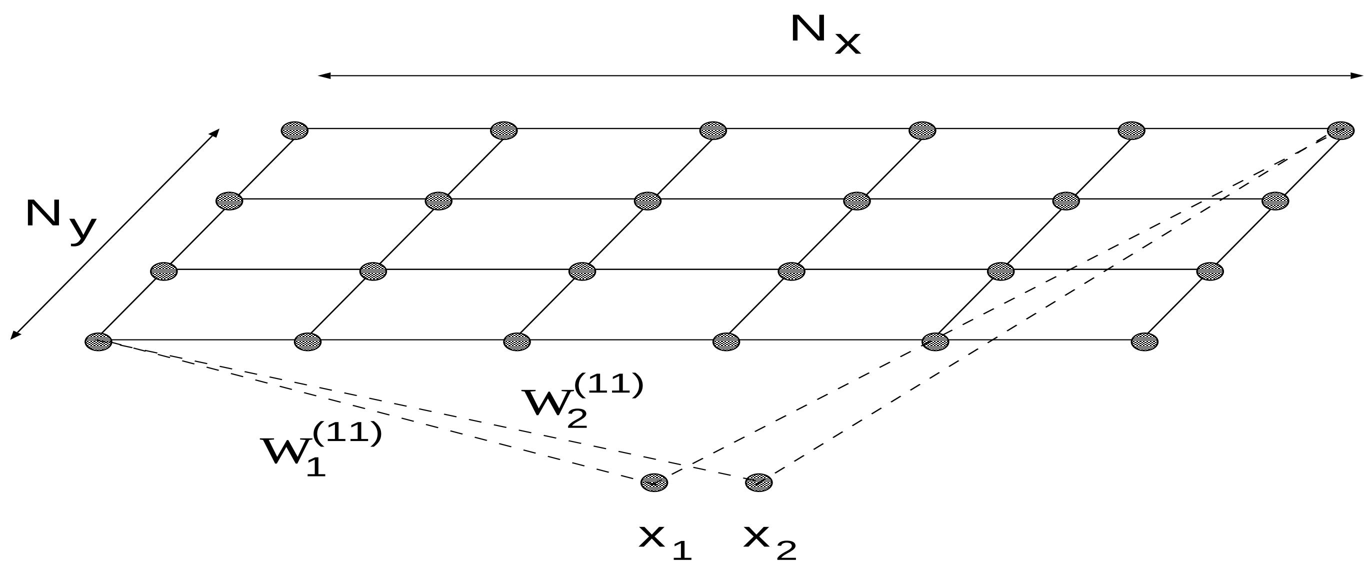
We now assume a Neural Network with two input neurons and $N_x \times N_y$ output neurons, no hidden layers.

The location of each of the utput neurons is denoted with

$$\mathbf{n} = (n_x, n_y) \quad n_x = 1, \cdots N_x, \text{ and } n_y = 1, \cdots N_y$$

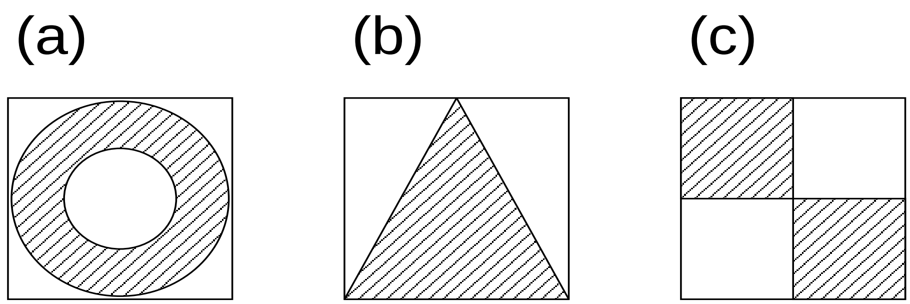
Each of the two input neurons connect to each of the output neurons via a synapse. The weight value of the synapse from the d-th input neuron ($d = 1, 2$) to the output neuron $\mathbf{n}$ is denoted as $w_d^{\mathbf{n}}$ and we use a notation

$$\mathbf{w}^{\mathbf{n}} = (w_1^{\mathbf{n}}, w_2^{\mathbf{n}})$$



Then the $\mathbf{w^n}$ can be self-organized in the following way:

1. Determin $\mathbf{w^n}$ at random
2. The input $\mathbf{x} = (x_1, x_2)$ is given by drawing the point (x_1, x_2) at random from a given domain, like



3. Select one output neuron $\mathbf{n}_0$ such that $||\mathbf{x} - \mathbf{n}_0||$ is minimum of all $\mathbf{x} - \mathbf{n}$.
4. Renew weight values as follows:

$$\mathbf{w^n} = \mathbf{w^n} + \epsilon \cdot g(\mathbf{n} - \mathbf{n}_0)(\mathbf{x} - \mathbf{w^n}), \tag{1}$$

where η decreases gradually as the numer of repetition (see Step 5. bellow) increases,

$$g(\mathbf{n} - \mathbf{n}_0) = \exp(-\frac{||\mathbf{x} - \mathbf{n}_0||^2}{2\delta}),$$

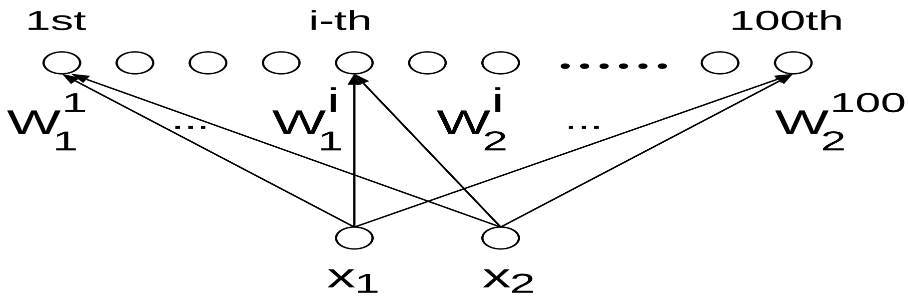
and δ is a constant to adjust to obtain an appropriate result.

Or,

$$g(\mathbf{n} - \mathbf{n}_0) \equiv 1.$$

5. Repeat 2. - 4. above.

Excersise 1 *As an example of mapping from 2D to 1D, construct a SOM network with two input and 100 output with $N_x = 100$ and $N_y = 1$.*



Then try the following algorithm.

1. Choose an input point by sampling it at randome from 2D triangle like (b) in the previous page,

2. Feed the cordinate of the point (x_1, x_2) to the input of the Kohonen SOM network, so that weight values are self-organized according to Eq.(1), that is,

$$w_d^i = w_d^i + \epsilon \cdot \exp\{-(x_d - w_d^{\max})^2\} \cdot (x_d - w_d^i)$$

   where $w_d^{\max}$ is the weight value from x_d to the output neuron of the winner, $d = 1, 2$ and $i = 1, 2, \cdots, 100$. The value of ϵ is decreased gradually like, for example,

   · Starting with $\epsilon = 0.05$ for the 1st repetion 1 – 5000; $\epsilon = 0.04$ for the 2nd repetion 5001 – 10000; $\epsilon = 0.03$ for the 3rd repetion 10001 – 15000; $\epsilon = 0.03$ for the 4th repetion 15001 – 20000; and $\epsilon = 0.01$ for the last repetion 20001 – 25000

   Or, simply by

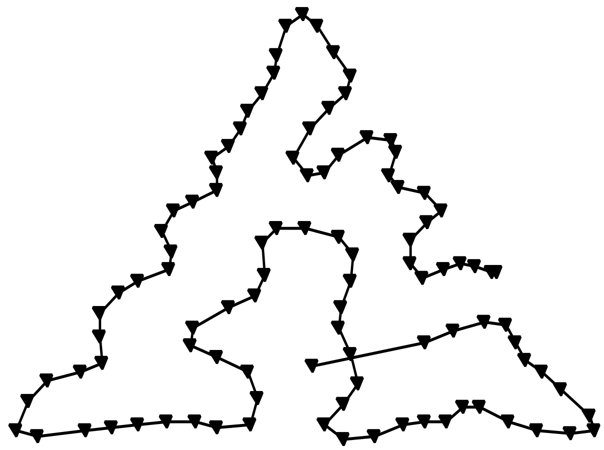
$$w_d^i = w_d^i + \epsilon(x_d - w_d^i)$$

3. Repeat 1. and 2. above, say, 25000 times.

4. Plot 100 pairs of weigthts (w_1^i, w_2^i) $\quad i = 1, 2, \cdots, 100$

The bellow is an example of the result of plotting

$$(w_1^1, w_2^1), (w_1^2, w_2^2), \cdots, (w_1^{100}, w_2^{100})$$



A more general discription of the Network

In the more general form, we also assume a two layered Neural Network. It has also $N_x \times N_y$ output neurons but the number of input neurons is more generally d. It maps a point in d-dimensional space to a point in two-dimensional space. Hence we might use it for the purpose of dimension reduction.