Kohonen's Self Organizing Map: Reduction of Dimension

Akira Imada

1 Let's construct a framework.

We can use the Kohonen's SOM for dimension reduction. It's important because, we, human being couldnt never imagine the space whose dimension is more than three. Here to study how it works, we map 2-dimensional space to 1-dimensional space. Though both worlds are visible but this is just to learn how it works. So the frame work is as Figure 1

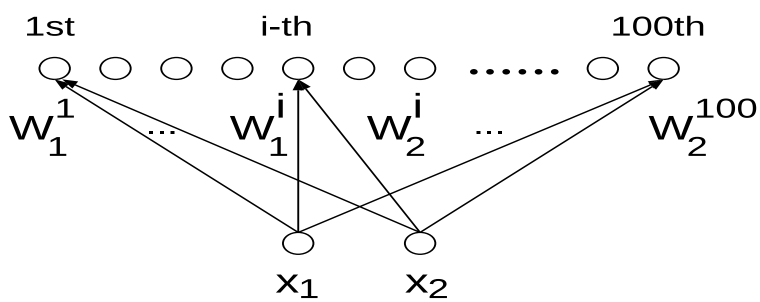


Figure 1: A framework of Kohonen's SOM for a mapping from 2-D to 1-D space.

Excersise 1 *As an example of mapping from 2D to 1D, construct a SOM network with two input and 100 output as follows:*

(1) All of the two input neurons connect to all of 100 output neurons. (So we have 200 syanapses)

(2) Give each of all these 200 syanapses a random weight value between 0.00 to 1.00.

(3) Design a man-machine interface to show Winner-Take-All scheme on the display screen

- *(i) Display 100 output neurons in a 10 times 10 rectanglar array on the display screen.*
- *(ii) Disign so that the algorithm can accept 2 input from key-board.*
- *(iii) Make only one neuron (= winner) highlited. The winner is the neuron which has the synaptic weights (w_1, w_2) closest to the input (x_1, x_2).*

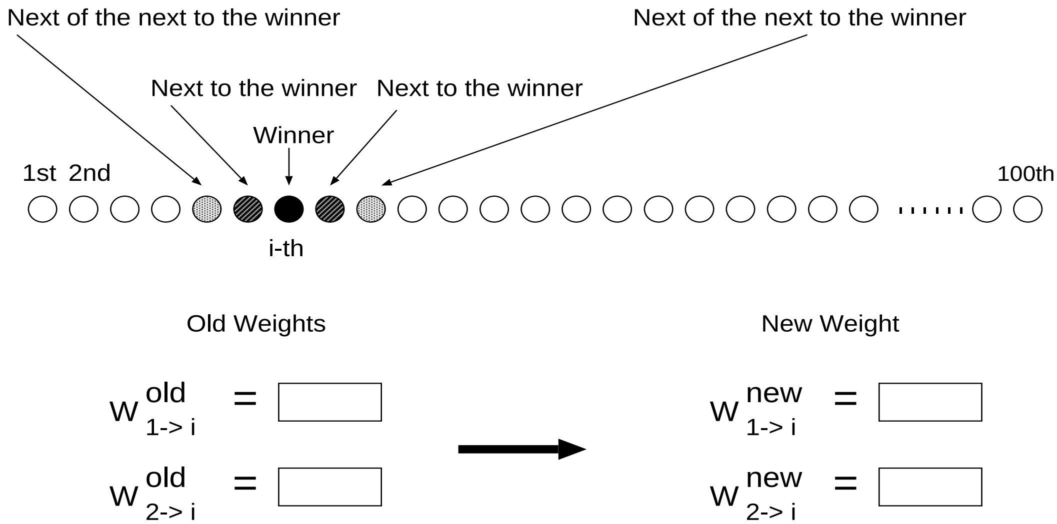


Figure 2: An example of the grphic on the display screen to show winner take all algorithm.

2 Let's Renew Weights – Winner Takes All

Algorithm 1 *As an example of mapping from 2D to 1D, construct a SOM network with two input and 100 output as follows:*

(0) Initialize all the weights at random from -1 to 1.

(1) t = 0.

(2) Pick up a point (x1, x2) randomly from inside a triangle in 2D space which is in the domain $[-1, 1]$

(3) Give the coordinate (x1, x2) of the point to the SOM as a set of inputs.

(4) Renew weights only of the winner neuron and the next neuron's. That is $2 \times 3 = 6$ *weights. with the equation*

$$w = w + E * K * (x - w) \tag{1}$$

where

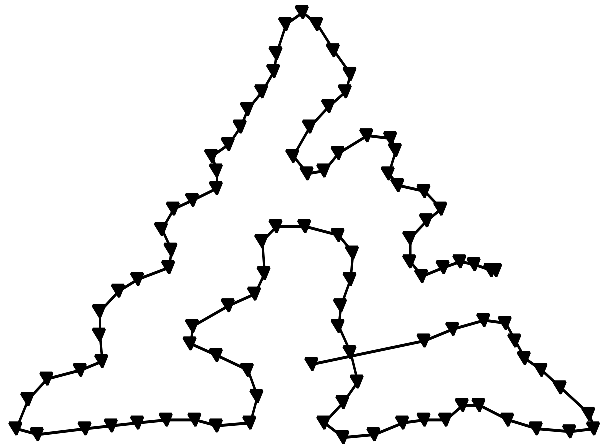
$$E = 0.0001(25000 - t) \tag{2}$$

so that learning-ratio is decreased gradually, and

$$K = \begin{cases} 1.00 & \text{if the neuron is the winner} \\ 0.50 & \text{if the neuron is the next to the winner} \\ 0.00 & \text{otherwise} \end{cases} \tag{3}$$

(5) t = t + 1

(6) Repeat (1) to (3) from t = 1 to t = 25000

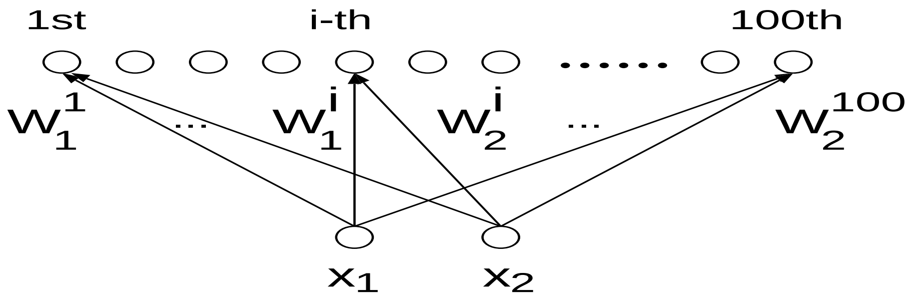


3 Let's observe a dimension reduction.

Excersise 2 *Let's observe a dimension reduction, with using SOM created via Excersise 1 in the following way.*

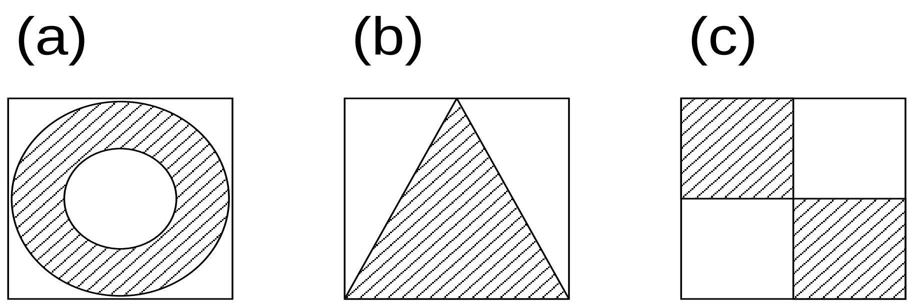
1. *Pick up 5 points inside the triangle used in Excersise 1 close to each of the three verteces.*
2. *Give each of the three groups a differnt color (e.g. R,G, and B).*
3. *Give these 15 points as inputs and give each of the 15 output neurons the corresponding color given in 2.*
4. *Show all the 100 output neurons with assigned colors if any.*

Excersise 3 *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) below.



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.(**??**), 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

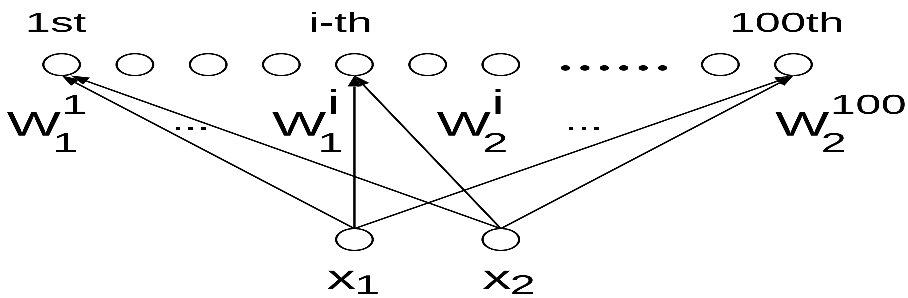
$$w_d^i = w_d^i + \epsilon(x_d - w_d^i)$$

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

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


1st
i-th
100th
w_1^1
...
w_1^i
w_2^i
...
w_2^{100}
x_1
x_2