

Kohonen Architecture

# The Kohonen Self-Organizing Map

Developed by Prof. Tuevo Kohonen of Helsinki University of Technology:

What is it?

1. A Kohonen feature map is composed of neurons competing for each other.
2. Each Neuron determines its output according to a weighted sum

Neuron output= Sum of weight vectors times its inputs  $\sum w_{ij} x_i$

3. Their weights and the inputs are normalized which means that the magnitude of the weight and input vectors are equal to one.
4. The neuron with the largest output is the winner; this neuron has a final output of 1. All neurons in the layer will have an output of zero.
5. Differing input patterns end up firing different winner neurons and same patterns classify to the same winning output neuron.

**Normalization of a Vector:** this is done in the generation of Weights in the input to Kohonen feature map layer

Ex:

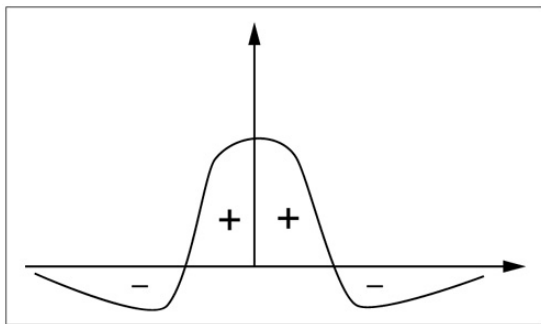
$A = x, y, z$

(Normalization)  $n = 1/\sqrt{x^2 + y^2 + z^2}$

$A' = xn, yn, zn$

## Lateral Inhibition

Lateral inhibition is a process that takes place in some biological neural networks. Lateral connections of neurons in a given layer are formed, and squash distant neighbors. The strength of connections is inversely related to distance. The positive, supportive connections are termed as excitatory while the negative, squashing connections are termed inhibitory.



The Mexican Hat Function

## Training Law for the Kohonen Map

The change in weight vector for a given output neuron is a gain constant  $\alpha$ , multiplied by the difference between input vector and the old weight vector


$$W_{\text{new}} = W_{\text{old}} + \alpha * (\text{Input} - W_{\text{old}})$$

Alpha is a gain constant between 0.01 and 1.

The importance of this law is to generate weights to its input (getting weights closer to the input via alpha constant) essentially memorizing the input data set classes.

## The Neighborhood Size and Alpha

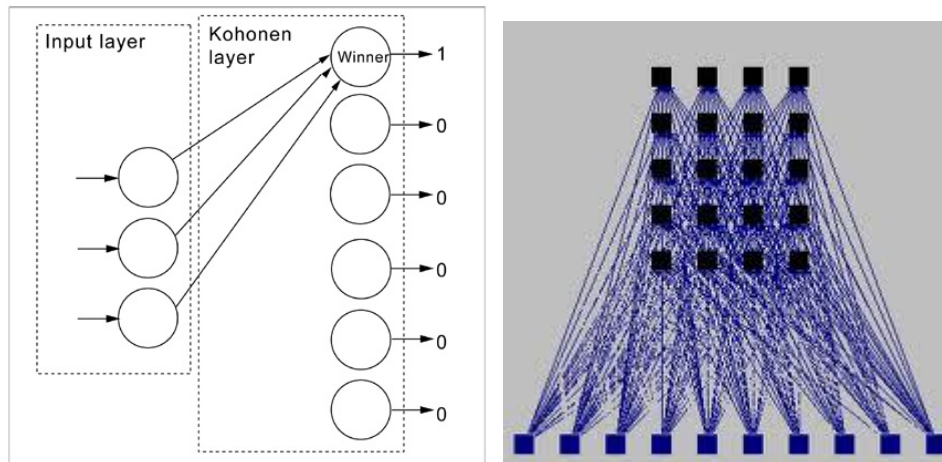


The Mexican hat, depicted as  specifies the distance of participation in training and weight vector changes. Those outside the distance do not participate. The system eventually is narrowed down to the single winning neuron.

Beside the neighborhood size, the alpha typically is also reduced during simulation (learning).

## The Kohonen Network

The Kohonen network has two layers; the input layer and the Kohonen output layer.

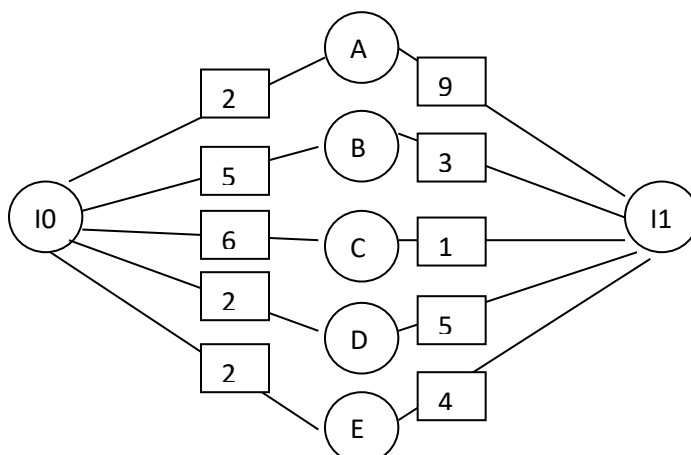


System Summary Procedure:

We model lateral inhibition and excitation by looking at the maximum output for the output neurons and making that output belong to a winner neuron. Other outputs are inhibited by setting their outputs to zero. Training, or weight update, is performed on all outputs that are within a neighborhood size distance from the winner neuron. Neurons outside the neighborhood do not participate in training.

Example: In a neural net of 2 input layer nodes I0 and I1 and 5 Kohonen Output nodes A, B, C, D and E.

Step 1: Randomize Weight Values (Better if random values are within the range of the input patterns)



Step 2: Normalize weight vectors

For I0:

$$N=1/\sqrt{(2^2+5^2+6^2+2^2+2^2)} = 0.1171$$

$$W0= 2*0.1171= 0.2342$$

$$W1=5* 0.1171=0.5855$$

$$W2=6* 0.1171=0.7026$$

$$W3=2* 0.1171=0.2342$$

$$W4=2*0.1171=0.2342$$

For I1:

$$N=1/\sqrt{(9^2+3^2+1^2+5^2+4^2)} = 0.0870$$

$$W5= 9*0.0870= 0.7832$$

$$W6=3* 0.0870=0.261$$

$$W7=1* 0.0870=0.0870$$

$$W8=5* 0.0870=0.435$$

$$W9=4*0.0870=0.348$$

Step 3: ACTIVATION: Calculate the nodes in the output layer

$$A= I0*W0+I1*W5$$

$$B= I0*W1+I1*W6$$

$$C= I0*W2+I1*W7$$

$$D= I0*W3+I1*W8$$

$$E= I0*W4+I1*W9$$

Step 3: Find the Neuron with the largest output value

Step 4: Change the value of the non winning output nodes to 0

Step 5: Update the Winning Neuron and its neighbors depending on the neighborhood size.

Example: if the winning neuron is C, and the neighborhood size is 2 and alpha is 0.5

Participating nodes:

$$W1=0.5855+0.5*(I0-0.5855)$$

$$W2=0.7026+0.5*(I0-0.7026)$$

$$W3=0.22342+0.5*(I0-0.22342)$$

$$W6=0.261+0.5*(I1-0.261)$$

$$W7=0.0870+0.5*(I1-0.0870)$$

$$W8=0.435+0.5*(I1-0.435)$$

Step 6: Get the Distance between the winner weight vector and the input vector (Euclidian Distance)

Example: if the winning neuron is C, and the neighborhood size is 2 and alpha is 0.5

$$(Distance\ at\ I0)\ D0=(I0-W2)$$

$$(Distance\ at\ I1)\ D1=(I1-W7)$$

$$\text{Dist} = \sqrt{D_0^2 + D_1^2}$$

Important:

Alpha is decremented every period (every cycle set of patterns) by 0.1 limit at 0

Neighborhood size is decrement every x number of periods (user or programmer discretion)

When does the SOM stop?

1. Until cycle ends in neighborhood size 0
2. Average distance pattern is  $\leq$  required distance (typically 0.05)  
 $\text{AveDistpat} = \text{total\_distance\_in\_cycle} / \text{number\_patterns\_per\_cycle}$