دانشگاه کردستان
University of Kurdistan
زانکۆی کوردستان

Department of Computer Engineering
University of Kurdistan

Neural Networks (Graduate level)

Competitive Networks, Clustering and unsupervised learning

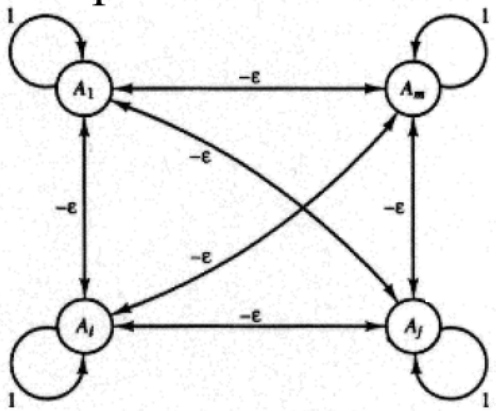
By: Dr. Alireza Abdollahpouri

Competitive networks

- Competition between neurons has been observed in biological nerve systems
- To classify an input pattern into one of the m classes
  - ideal case: one class node has output 1, all other 0 ;
  - often more than one class nodes have non-zero output

Fixed-weight Competitive Nets

- **Maxnet**
  - Lateral inhibition between competitors



$$\text{weights} : w_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\varepsilon & \text{otherwise} \end{cases}$$

$$\text{activation function} : f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

- Notes:
  - Competition:
    - iterative process until the net stabilizes (at most one node with positive activation)
  - $0 < \varepsilon < 1/m$, where m is the # of competitors
    - ε too small: takes too long to converge
    - ε too big: may suppress the entire network (no winner)

Maxnet

- Consider the action of a MAXNET with four neurons and inhibitory weights ε = 0.2.

$$a_1(0) = 0{,}2 \qquad a_2(0) = 0{,}4 \qquad a_3(0) = 0{,}6 \qquad a_4(0) = 0{,}8$$

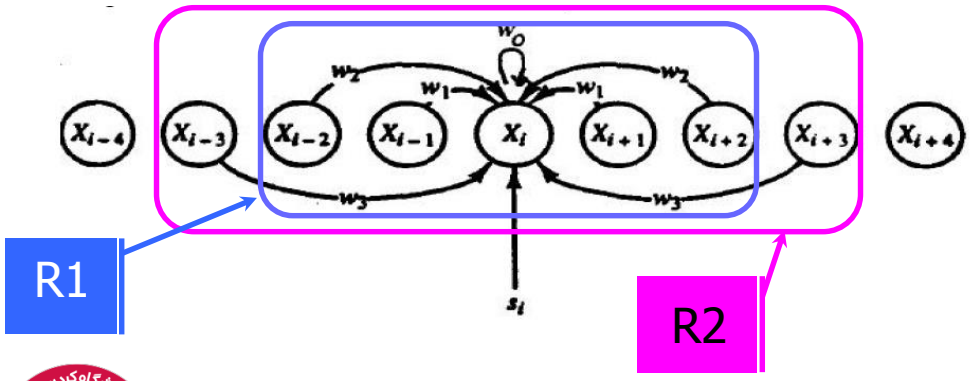
$$a_j(new) = f\left[a_j(old) - \varepsilon \sum a_k(old)\right]$$

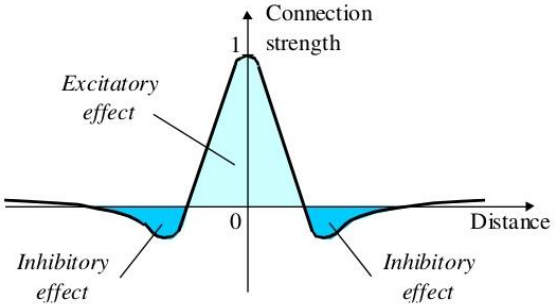
$a_1(1) = 0{,}0$	$a_2(1) = 0{,}08$	$a_3(1) = 0{,}32$	$a_4 = 0{,}56$
$a_1(2) = 0{,}0$	$a_2(2) = 0{,}0$	$a_3(2) = 0{,}192$	$a_4 = 0{,}48$
$a_1(3) = 0{,}0$	$a_2(3) = 0{,}0$	$a_3(3) = 0{,}096$	$a_4 = 0{,}442$
$a_1(4) = 0{,}0$	$a_2(4) = 0{,}0$	$a_3(4) = 0{,}008$	$a_4 = 0{,}442$
$a_1(5) = 0{,}0$	$a_2(5) = 0{,}0$	$a_3(5) = 0{,}0$	$a_4 = 0{,}421$

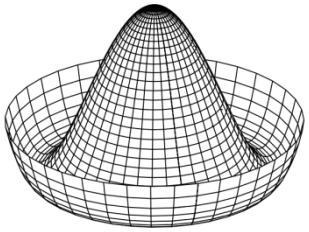
If more than one node has a nonzero activation, continue; otherwise, stop.

Mexican hat

- The Mexican Hat network [Kohonen, 1989a] is a more general contrast enhancing subnet than the MAXNET. Three types of links can be found in such network:
  - close neighbors: cooperative (mutually excitatory , w > 0)
  - farther away neighbors: competitive (mutually inhibitory, w < 0)
  - too far away neighbors: irrelevant (w = 0)







Mexican hat

R2 Radius of region of interconnections; X_i is connected to units X_{i+k} and X_{i-k} for k = 1, ... ,*R2*·

R1 Radius of region with positive reinforcement; $R1 < R2$.

$\mathbf{w_k}$ Weight on interconnections between X_i and units X_{i+k} and X_{i-k} :

w_k is positive for $0 \leq k \leq R1$

w_k is negative for $R1 < k \leq R2$

x Vector of activations.

x_old Vector of activations at previous time step.

t_max Total number of iterations of contrast enhancement.

s External signal.

Mexican hat- Algorithm

Step 0: Initialize parameters t_max, R1, R2 as desired Initialize weights:

$$w_{ij} = \begin{cases} C_1 > 0 & \text{for} \quad k = 0, \ldots R_1 \\ C_2 < 0 & \text{for} \quad k = R_1 + 1, \ldots R_2 \end{cases}$$

Initialize **x_old** to **0**

Step 1: Present external signal **s**: x=**s**

Save activation in array x_old (for i=1,…,n) → $x\_old_i = x_i \qquad i = 1,\ldots,n$

Set iteration counter: t = 1

Step 2: While t is less than t_max, do steps 3-7

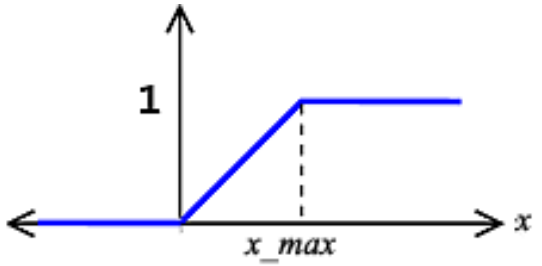
Mexican hat- Algorithm

Step 3: Compute net input (i = 1, …n)

$$x_i = C_1 \sum_{k=-R_1}^{R_1} x\_old_{i+k} + C_2 \sum_{k=-R_2}^{-R_1-1} x\_old_{i+k} + C_2 \sum_{k=-R_1+1}^{R_2} x\_old_{i+k}$$

Step 4: Apply activation function f (ramp from 0 to x_max, slope 1):

$$f(x) = \begin{cases} 0 & if \quad x < 0 \\ x & if \quad 0 \le x \le \max \\ x\_\max & if \quad x > \max \end{cases}$$



$$x_i = \min(x\_\max, \max(0, x_i)) \quad i = 1,\ldots,n$$

Mexican hat- Algorithm

Step 5: Save current activations in x_old:

$$x\_old_i = x_i \qquad i = 1,\ldots,n$$

Step 6: Increment iteration counter: t=t+1

Step 7: Test stopping condition:
If t < t_max, continue; otherwise, stop.

Mexican hat- Example

- Using the Mexican Hat Algorithm We illustrate the Mexican Hat algorithm for a simple net with seven units. The activation function for this net is:

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \le x \le 2 \\ 2 & \text{if } x > 2 \end{cases}$$

$R1 = 1,\ R2 = 2$

$C1 = 0.6,\ C2 = -0.4$

The external signal s is (0.0, 0.5, 0.8, 1.0, 0.8, 0.5, 0.0)

Sol:

$x_i = s_i$

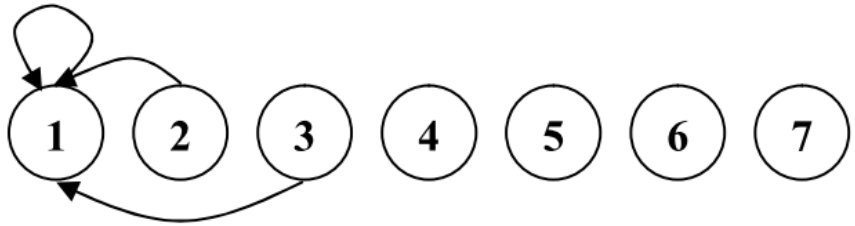
x = (0.0, 0.5, 0.8, 1.0, 0.8, 0.5, 0.0).

Mexican hat- Example

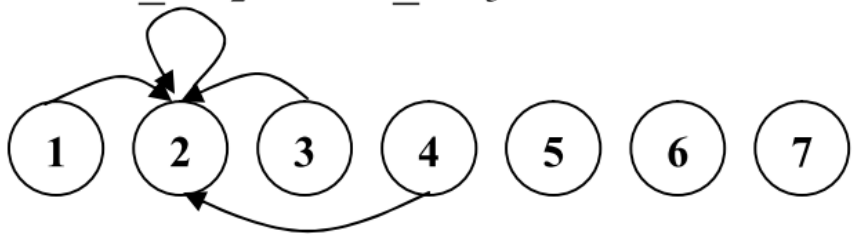
Save in x_old:

x_old = (0.0, 0.5, 0.8, 1.0, 0.8, 0.5, 0.0).

$$xi = C_1 \sum_{k=-R1}^{R1} x\_old_{i+k} + C_2 \sum_{k=-R2}^{-R-1} x\_old_{i+k} + C_2 \sum_{k=R1+1}^{R2} x\_old_{i+k}$$



$x_1 = 0.6\ x\_old_1 + 0.6\ x\_old_2 - 0.4\ x\_old_3$



$x_2 = 0.6\ x\_old_1 + 0.6\ x\_old_2 + 0.6\ x\_old_3 - 0.4\ x\_old_4$

Mexican hat- Example

$$x_3 = -0.4x\_old_1 + 0.6x\_old_2 + 0.6x\_old_3 + 0.6x\_old_4 - 0.4x\_old_5$$

$$x_4 = -0.4x\_old_2 + 0.6x\_old_3 + 0.6x\_old_4 + 0.6x\_old_5 - 0.4x\_old_6$$

$$x_5 = -0.4x\_old_3 + 0.6x\_old_4 + 0.6x\_old_5 + 0.6x\_old_6 - 0.4x\_old_7$$

$$x_6 = -0.4\ x\_old_4 + 0.6\ x\_old_5 + 0.6\ x\_old_6 + 0.6\ x\_old_7$$

$$x_7 = -0.4\ x\_old_5 + 0.6\ x\_old_6 + 0.6\ x\_old_7$$

Mexican hat- Example

$\underline{t = 1}$

$x_1 = 0.6(0.0) + 0.6(0.5) - 0.4(0.8) = -0.2$

$x_2 = 0.6(0.0) + 0.6(0.5) + 0.6(0.8) - 0.4(1.0) = 0.38$

$x_3 = -0.4(0.0) + 0.6(0.5) + 0.6(0.8) + 0.6(1.0) - 0.4(0.8) = 1.06$

$x_4 = -0.4(0.5) + 0.6(0.8) +,0.6(1.0) + 0.6(0.8) - 0.4(0.5) = 1.16$

$x_5 = -0.4(0.8) + 0.6(1.0) + 0.6(0.8) + 0.6(0.5) - 0.4(0.0) = 1.06$

$x_6 = -0.4(1.0) + 0.6(0.8) + 0.6(0.5) + 0.6(0.0) = 0.38$

$x_7 = -0.4(0.8) + 0.6(0.5) + 0.6(0.0) = -0.2.$

$x = (0.0, 0.38, 1.06, 1.16, 1.06, 0.38, 0.0).$

Mexican hat- Example

***t* = 2**

$x_1 = 0.6(0.0) + 0.6(0.38) - 0.4(1.06) = -0.196$

$x_2 = 0.6(0.0) + 0.6(0.38) + 0.6(1.06) - 0.4(1.16) = 0.39$

$x_3 = -0.4(0.0) + 0.6(0.38) + 0.6(1.06) + 0.6(1.16) - 0.4(1.06) = 1.14$

$x_4 = -0.4(0.38) + 0.6(1.06) + 0.6(1.16) + 0.6(1.06) - 0.4(0.38) = 1.66$

$x_5 = -0.4(1.06) + 0.6(1.16) + 0.6(1.06) + 0.6(0.38) - 0.4(0.0) = 1.14$

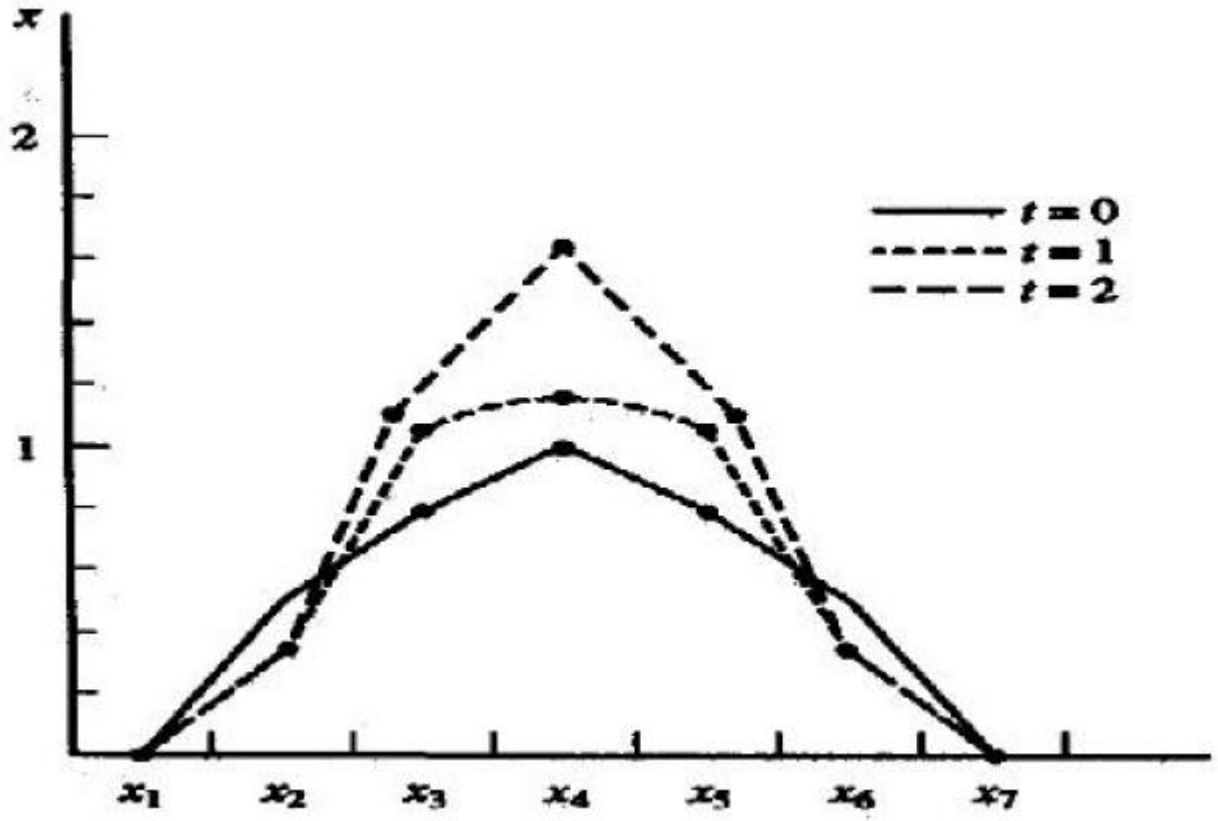
$x_6 = -0.4(1.16) + 0.6(1.06) + 0.6(0.38) + 0.6(0.0) = 0.39$

$x_7 = -0.4(1.06) + 0.6(0.38) + 0.6(0.0) = -0.196$

x = (0.0, 0.39, 1.14, 1.66, 1.14, 0.39, 0.0).

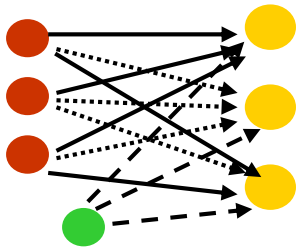
The pattern of activations is shown for *t* = 0, 1, and *2* in Figure 3.

Mexican hat- Example



Hamming Network

A Hamming net [Lippmann, 1987; DARPA, 1988] is a maximum likelihood classifier net that can be used to determine which of several exemplar vectors is most similar to an input vector (an n-tuple)

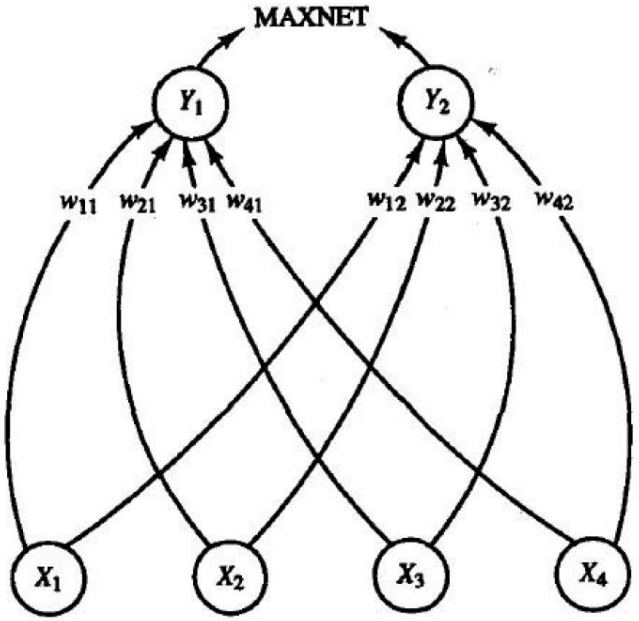


Hamming distance of two vectors is the number of components in which the vector differ.

$$H\left(x_1=\begin{bmatrix}0\\1\\1\end{bmatrix},x_2=\begin{bmatrix}1\\1\\0\end{bmatrix}\right)=2 \quad \Rightarrow \quad \bar{H}=\frac{H}{n(=\#\,of\ \ dimentions)}=\frac{2}{3}$$

Hamming Network

- By setting the weights to be one-half the exemplar vector and setting the value of the bias to n/2, the net will find the unit with the closest exemplar simply by finding the unit with the largest net input.



Hamming network-example

A Hamming net to cluster four vectors Given the exemplar vectors: e(1) = (1, -1, -1, -1,), e(2) = (-1, -1, -1, 1)

Using the Hamming net to find the exemplar that is closest to each of the bipolar input pattern:

(1, 1, -1, -1), (1, -1, -1, -1), (-1, -1, -1, 1), and (-1, -1, 1, 1)

Hamming network-example

Store the m exemplar vectors in the weights:

$$W = \begin{bmatrix} .5 & -.5 \\ -.5 & -.5 \\ -.5 & -.5 \\ -.5 & -.5 \end{bmatrix} \qquad \text{where } w_{ij} = \frac{e_i(j)}{2}$$

Initialize the biases :

$$b_1 = b_2 = 2 = n/2$$

n = number of input nodes = 4 (in this case)

Hamming network-example

For the vector **x**=(1,1, -1, -1), let's do the following steps:

$$y\_in_1 = b_1 + \sum_i x_i w_{i1} = 2 + 1 = 3$$

$$y\_in_2 = b_2 + \sum_i x_i w_{i2} = 2 - 1 = 1$$

These values represent the Hamming similarity because (1, 1, -1, -1) agrees with **e**(1)= (1,-1, -1, -1) in the 1st , 3rd, and 4th components and because (1,1, -1, -1) agrees with **e**(2)=(-1,-1,-1,1) in only the 3rd component

$$y_1(0) = 3$$
$$y_2(0) = 1$$

Since $y_1(0) > y_2(0)$, Maxnet will find that unit Y_1 has the best match exemplar for input vector **x** = (1, 1, -1, -1).

Hamming network-example

For the vector **x**=(1,-1, -1, -1), let's do the following steps:

$$y\_in_1 = b_1 + \sum_i x_i w_{i1} = 2 + 2 = 4$$

$$y\_in_2 = b_2 + \sum_i x_i w_{i2} = 2 + 0 = 2$$

Note that the input agrees with **e**(1) in all 4 components and agrees with **e**(2) in the 2nd and 3rd components

$$y_1(0) = 4$$
$$y_2(0) = 2$$

Since $y_1(0) > y_2(0)$, Maxnet will find that unit Y_1 has the best match exemplar for input vector **x** = (1, -1, -1, -1).

Hamming network-example

For the vector **x**=(-1,-1, -1, 1), let's do the following steps:

$$y\_in_1 = b_1 + \sum_i x_i w_{i1} = 2 + 0 = 2$$

$$y\_in_2 = b_2 + \sum_i x_i w_{i2} = 2 + 2 = 4$$

Note that the input agrees with **e**(1) in the 2nd and 3rd components and agrees with **e**(2) in all four components

$$y_1(0) = 2$$
$$y_2(0) = 4$$

Since $y_2(0) > y_1(0)$, Maxnet will find that unit Y_2 has the best match exemplar for input vector **x** = (-1, -1, -1, 1).

Hamming network-example

For the vector **x**=(-1,-1, 1, 1), let's do the following steps:

$$y\_in_1 = b_1 + \sum_i x_i w_{i1} = 2 - 1 = 1$$

$$y\_in_2 = b_2 + \sum_i x_i w_{i2} = 2 + 1 = 3$$

Note that the input agrees with **e**(1) in the 2nd component and agrees with **e**(2) in the 1st, 2nd, and 4th components

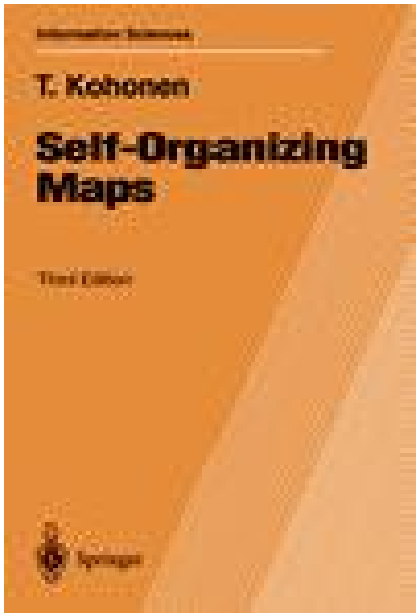
$$y_1(0) = 1$$
$$y_2(0) = 3$$

Since $y_2(0) > y_1(0)$, Maxnet will find that unit Y_2 has the best match exemplar for input vector **x** = (-1, -1, 1, 1).

Self Organized Map (SOM)

Proposed by: Teuvo Kohonen, Finland

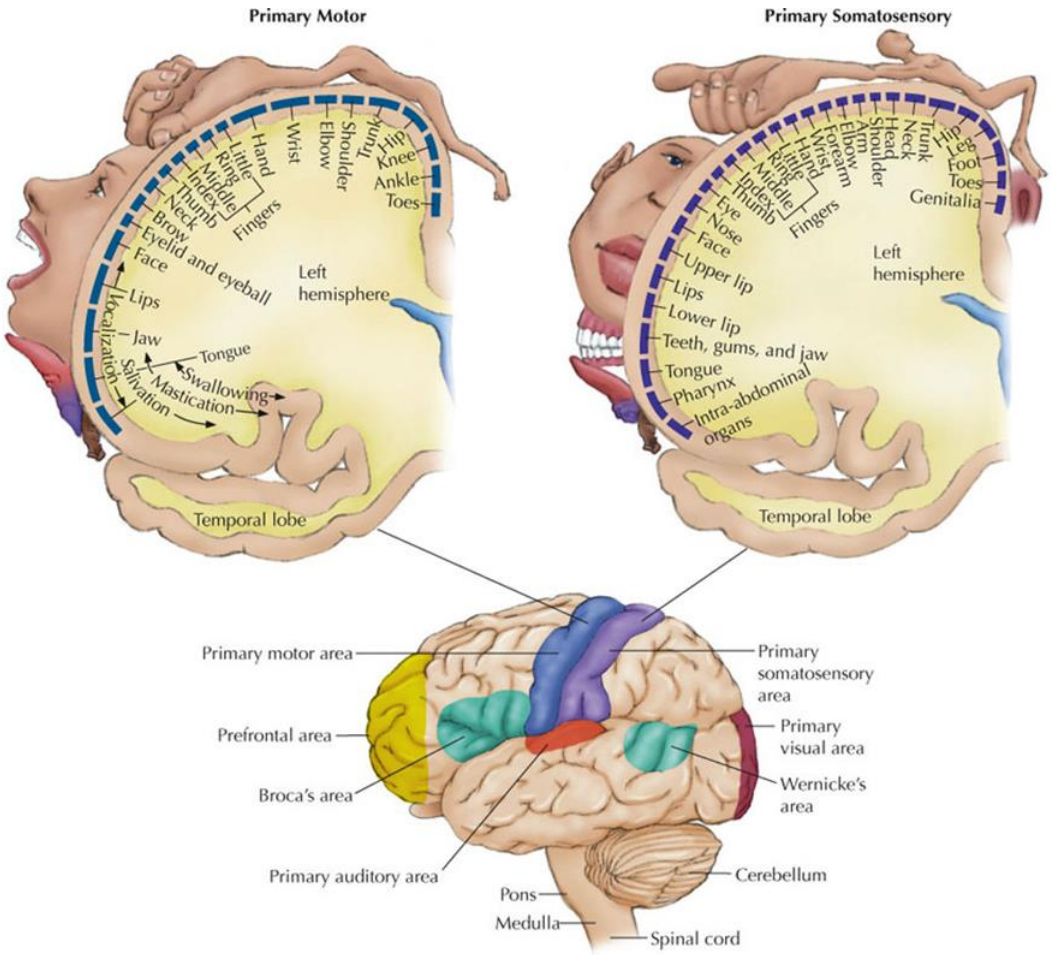
Author of the book: "Self-organizing maps", Springer (1997)



SOM assume a topological structure among the cluster units

This property is observed in the brain, but not found in other ANNs

SOM: Inspiration



Neurons dealing with closely related pieces of information are close together so that they can interact via short synaptic connections.

"Almost every region of the body is represented by a corresponding region in both the primary motor cortex and the somatic sensory cortex" (Geschwind 1979:106).

SOM: Inspiration

- Retinotopy: in the visual system, adjacent spots on the retina are represented by adjacent neurons in the lateral geniculate nucleus and the primary visual cortex.
- Tonotopy: in the auditory system, tones close to each other in terms of frequency are represented in topologically neighboring regions in the brain.

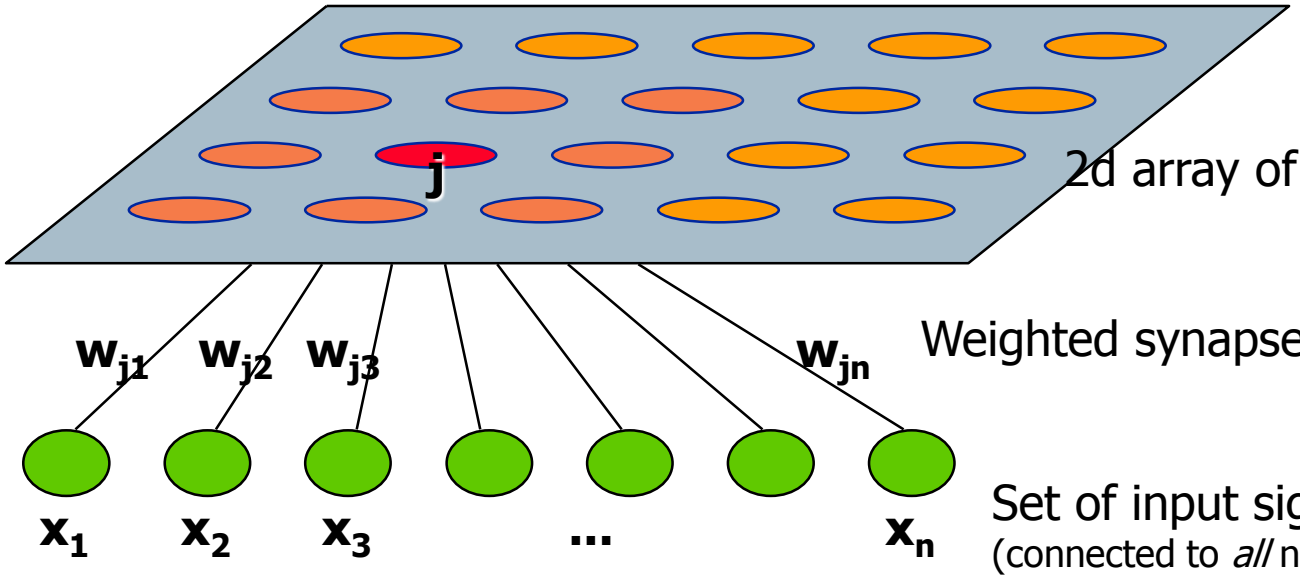
SOM: Inspiration

"small human"
-schematization of body in primary motor and sensory cortices

"The finger tips of humans have the highest density of receptors: about **2500 per square cm**!" (Kandel and Jessell 1991:374).

The bigger areas like your lips and hands represented here are the most sensitive.

SOM-Structure

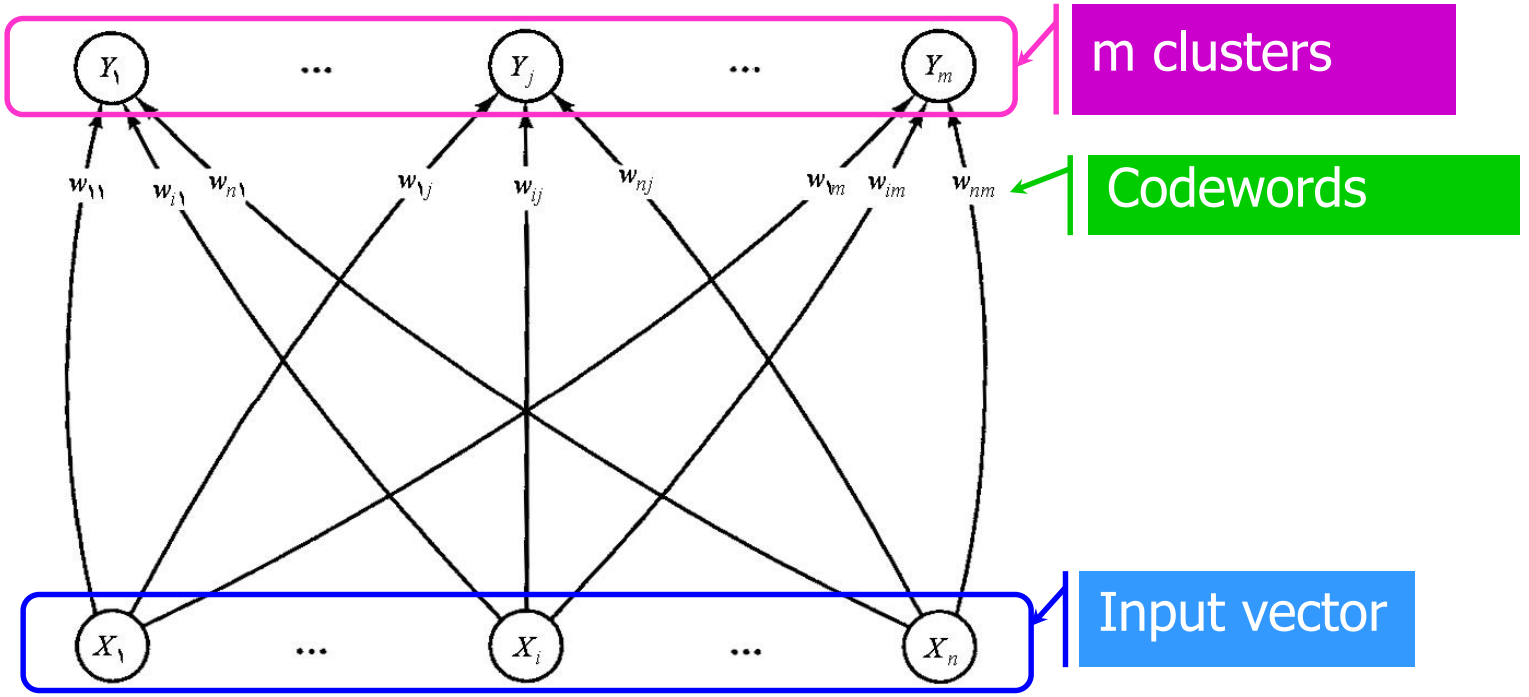


2d array of neurons

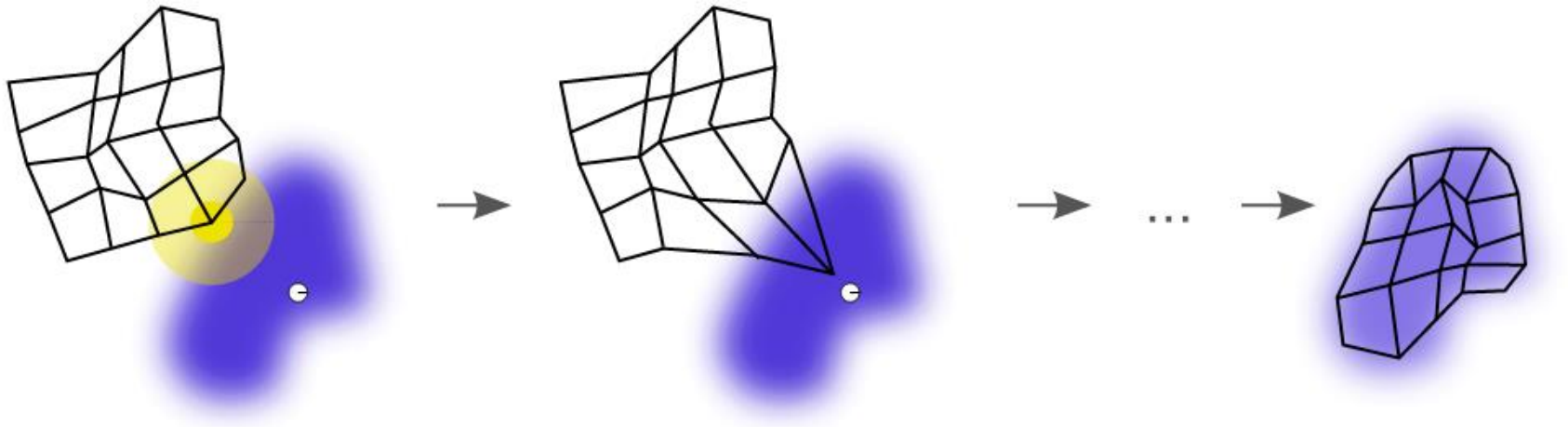
Weighted synapses

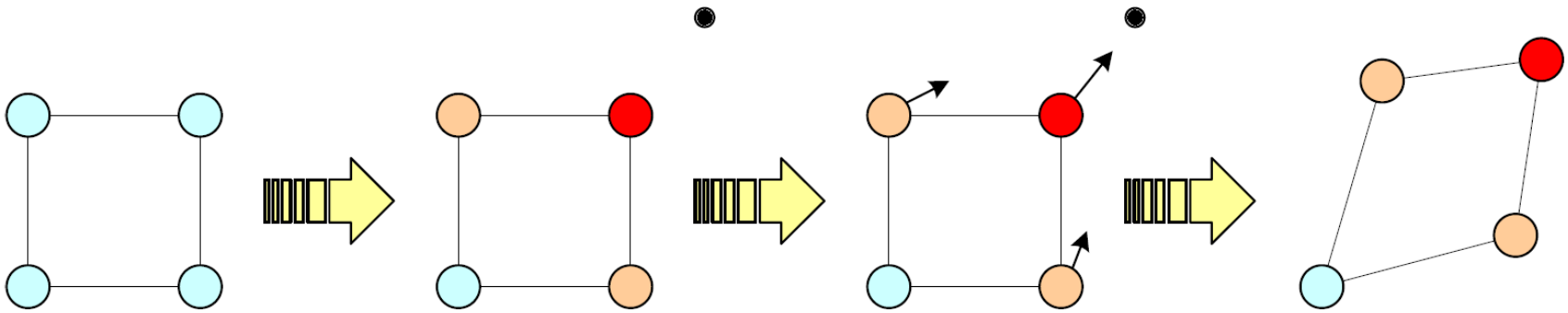
Set of input signals
(connected to *all* neurons in lattice)

SOM-Structure



SOM- Concept





SOM -Algorithm

- Initialization:

  1- Weights are initialized with small random values.

  2- Set topological neighborhood parameters

  3-Set learning rate parameters

- Repeat for a certain number of epochs **t** or until the map stops changing:

  For each input instance:

  1: **Competition**: Find the neuron whose weights are most similar to the input instance (winner). The neurons are competing to be the winner.

  2: **Cooperation**: Determine the neighborhood of the winner neuron, in which all neurons should be excited for being reasonably similar to the input instance. These neurons can be seen as cooperating with each other.

  3: **Adaptation**: Update the winner and its neighbors so that they are even more similar to the input instance.

SOM –Algorithm (Competition)

- In the learning algorithm, we have to find the neuron whose weights are most similar to the input instance (**winner**). How to determine the winner?

- **Similarity** (euclidean distance): $D(j) = \sum_i (w_{ij} - x_i)^2$

$W_j = (w_{j1}, w_{j2}, ..., w_{jn})$ are neuron j's weights and $X = (x_1, x_2, ..., x_n)$ is the input instance.

Neuron i with the minimum distance is the winner

SOM –Algorithm (Cooperation)

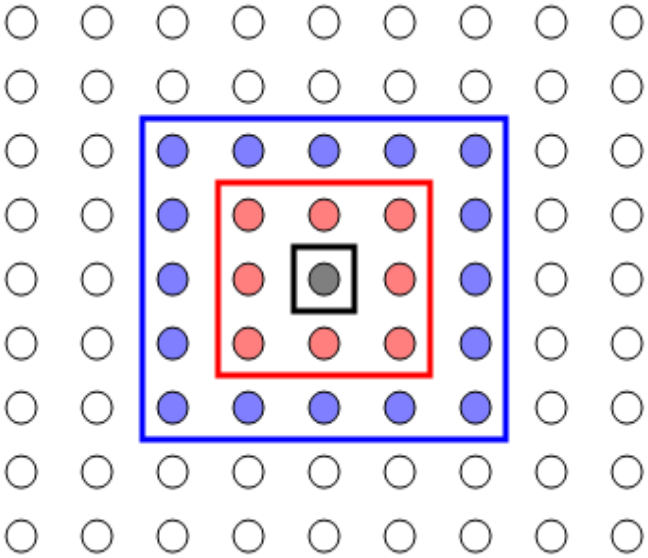
- According to neurobiological studies, similar neurons are close to each other in the cortex. When a neuron fires, its neighbors are often excited.
- We also want our self-organizing map to have similar neurons close to each other
- How to do that?

In SOM, not only the winning neuron will have its weights updated, but also its neighbors, although not as much as the winning neuron.

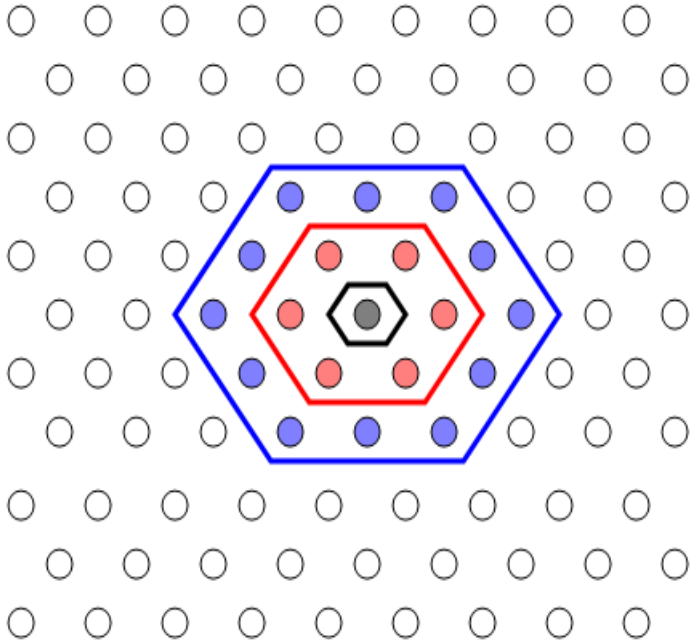
- The size of the neighborhood should shrink over time.

SOM- Neighborhood definition

linear array

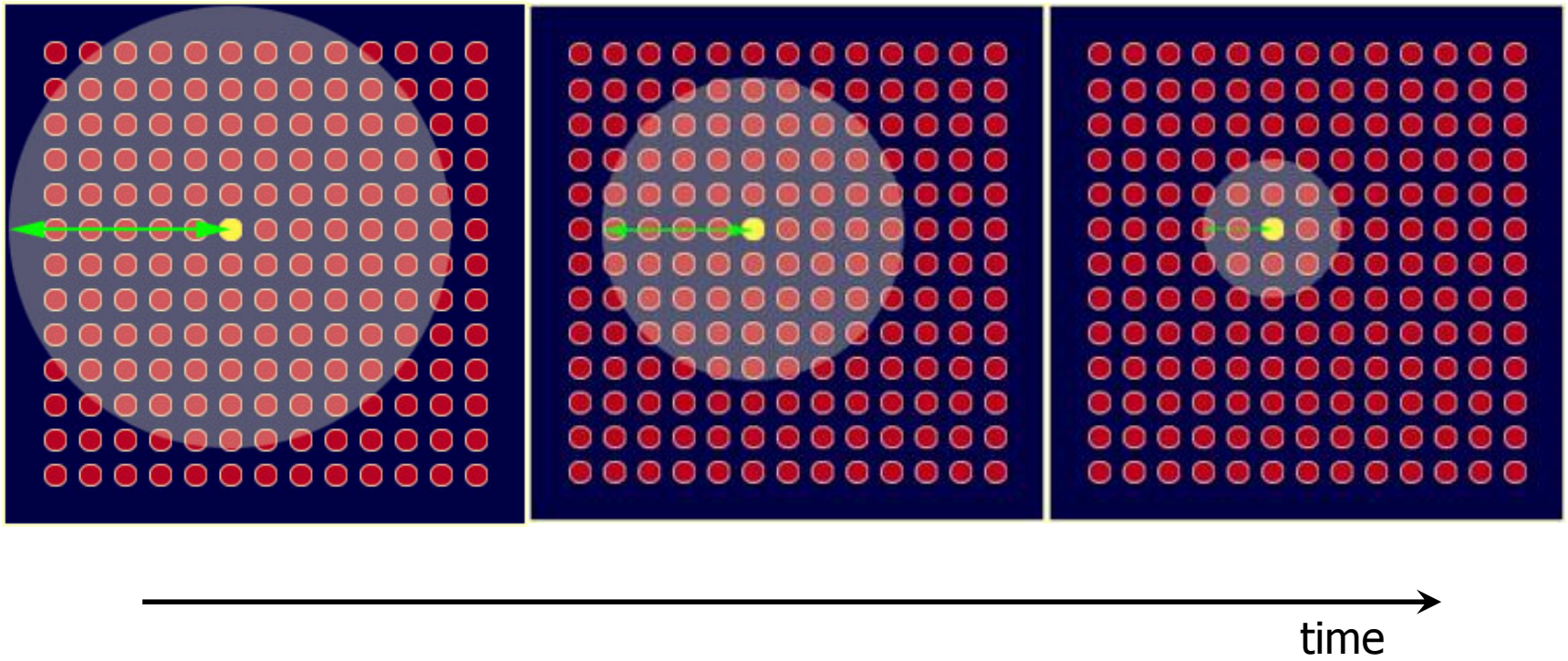


rectangular matrix of cluster



hexagonal matrix of cluster

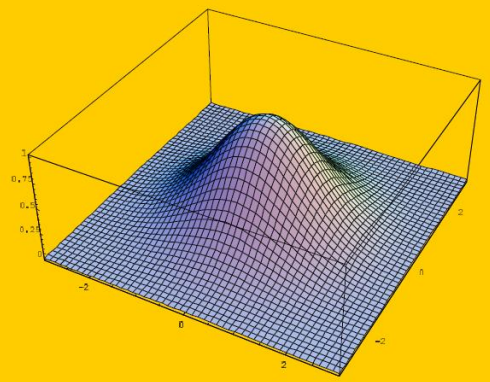
Neighbourhood



Common Neighbourhood functions

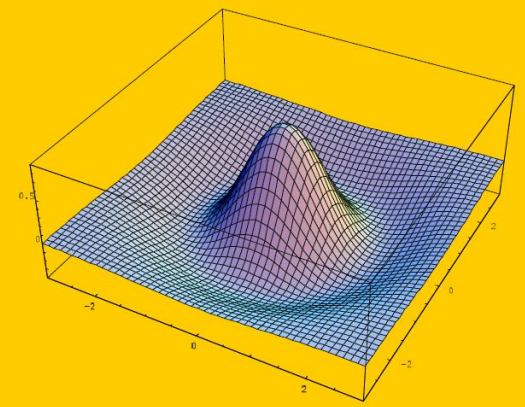
Gaußglocke mit Radius σ

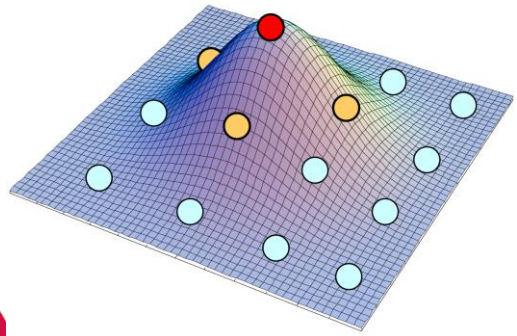
$$h_{r'}(r) = e^{-\frac{|r-r'|^2}{2\sigma^2}}$$



Mexican Hat mit Radius σ

$$h_{r'}(r) = (\frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}})(1-|r-r'|^2)e^{-\frac{|r-r'|^2}{2\sigma^2}}$$





It is maximal at the winning neuron, it is symmetrical about that neuron, it decreases monotonically to zero as the distance goes to infinity, and it is translation invariant (independent of the location of the winning neuron).

SOM- Adaptation

Weights should be updated in such a way that the new weights will be more similar (closer in distance) to the input.

$$w_{ij}(new) = w_{ij}(old) + \alpha[x_i - w_{ij}(old)]$$

Learning rate

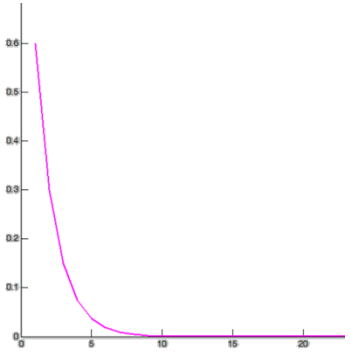
- Input X larger than weight Wj , increase Wj
- Input X smaller than weight Wj , decrease Wj

SOM-Example

A Kohonen SOM to cluster 4 vectors: $(1,1,0,0)$; $(0,0,0,1)$; $(1,0,0,0)$; $(0,0,1,1)$

The maximum number of clusters to be formed is m=2 (4 input nodes, 2 output nodes)

Suppose the learning rate is $\alpha(0)=0{,}6, \qquad \alpha(t+1)=0{,}5\,\alpha(t)$



No neighborhood

Random

$$\mathbf{w}=\begin{bmatrix} 0{,}2 & 0{,}8 \\ 0{,}6 & 0{,}4 \\ 0{,}5 & 0{,}7 \\ 0{,}9 & 0{,}3 \end{bmatrix} \qquad R=0 \qquad \alpha(0)=0{,}6$$

Initialization

SOM-Example

$(1, 1, 0, 0);$ $(0, 0, 0, 1);$ $(1, 0, 0, 0);$ $(0, 0, 1, 1)$

$$\mathbf{w} = \begin{bmatrix} 0{,}2 & 0{,}8 \\ 0{,}6 & 0{,}4 \\ 0{,}5 & 0{,}7 \\ 0{,}9 & 0{,}3 \end{bmatrix}$$

Distance calculation:

$$D(1) = (0{,}2-1)^2 + (0{,}6-1)^2 + (0{,}5-0)^2 + (0{,}9-0)^2 = 1{,}86$$
$$D(2) = (0{,}8-1)^2 + (0{,}4-1)^2 + (0{,}7-0)^2 + (0{,}3-0)^2 = 0{,}98$$

The i/p vector is closest to o/p node 2, so $J = 2$

The weights on the winning unit are updated ⇨ $w_{i2}(new) = w_{2i}(old) + 0{,}6[x_i - w_{i2}(old)] = 0{,}4w_{i2}(old) + 0{,}6x_i$

$$\mathbf{w} = \begin{bmatrix} 0{,}2 & 0{,}92 \\ 0{,}6 & 0{,}76 \\ 0{,}5 & 0{,}28 \\ 0{,}9 & 0{,}12 \end{bmatrix}$$

SOM-Example

$(1, 1, 0, 0)$; $(0, 0, 0, 1)$; $(1, 0, 0, 0)$; $(0, 0, 1, 1)$

$$\mathbf{w} = \begin{bmatrix} 0,2 & 0,8 \\ 0,6 & 0,4 \\ 0,5 & 0,7 \\ 0,9 & 0,3 \end{bmatrix}$$

Distance calculation:

$$D(1) = (0,2-0)^2 + (0,6-0)^2 + (0,5-0)^2 + (0,9-1)^2 = 0,66$$

$$D(2) = (0,92-0)^2 + (0,76-0)^2 + (0,28-0)^2 + (0,12-1)^2 = 2,2768$$

The i/p vector is closest to o/p node 1, so $J = 1$

The weights on the winning unit are updated ⇨

$$\mathbf{w} = \begin{bmatrix} 0,08 & 0,92 \\ 0,24 & 0,76 \\ 0,20 & 0,28 \\ 0,96 & 0,12 \end{bmatrix}$$

SOM-Example

$(1, 1, 0, 0)$; $(0, 0, 0, 1)$; $(1, 0, 0, 0)$; $(0, 0, 1, 1)$

$$\mathbf{w} = \begin{bmatrix} 0{,}2 & 0{,}8 \\ 0{,}6 & 0{,}4 \\ 0{,}5 & 0{,}7 \\ 0{,}9 & 0{,}3 \end{bmatrix}$$

Distance calculation:

$$D(1) = (0{,}08 - 1)^2 + (0{,}24 - 0)^2 + (0{,}2 - 0)^2 + (0{,}96 - 1)^2 = 1{,}8656$$

$$D(2) = (0{,}92 - 1)^2 + (0{,}76 - 0)^2 + (0{,}28 - 0)^2 + (0{,}12 - 0)^2 = 0{,}6768$$

The i/p vector is closest to o/p node 2, so $J = 2$

The weights on the winning unit are updated ⇒

$$\mathbf{w} = \begin{bmatrix} 0{,}08 & 0{,}968 \\ 0{,}24 & 0{,}304 \\ 0{,}20 & 0{,}112 \\ 0{,}96 & 0{,}048 \end{bmatrix}$$

SOM-Example

$(1, 1, 0, 0)$; $(0, 0, 0, 1)$; $(1, 0, 0, 0)$; $(0, 0, 1, 1)$

$$\mathbf{w} = \begin{bmatrix} 0{,}2 & 0{,}8 \\ 0{,}6 & 0{,}4 \\ 0{,}5 & 0{,}7 \\ 0{,}9 & 0{,}3 \end{bmatrix}$$

Distance calculation:

$$D(1) = (0{,}08 - 0)^2 + (0{,}24 - 0)^2 + (0{,}20 - 1)^2 + (0{,}96 - 1)^2 = 0{,}7056$$

$$D(2) = (0{,}968 - 0)^2 + (0{,}304 - 0)^2 + (0{,}112 - 1)^2 + (0{,}048 - 1)^2 = 2{,}724$$

The i/p vector is closest to o/p node 1, so $J = 1$

The weights on the winning unit are updated ⇨

$$\mathbf{w} = \begin{bmatrix} 0{,}032 & 0{,}968 \\ 0{,}096 & 0{,}304 \\ 0{,}680 & 0{,}112 \\ 0{,}984 & 0{,}048 \end{bmatrix}$$

End of first iteration

SOM-Example

$(1, 1, 0, 0)$; $(0, 0, 0, 1)$; $(1, 0, 0, 0)$; $(0, 0, 1, 1)$

$$\mathbf{w} = \begin{bmatrix} 0{,}032 & 0{,}968 \\ 0{,}096 & 0{,}304 \\ 0{,}680 & 0{,}112 \\ 0{,}984 & 0{,}048 \end{bmatrix}$$

Second Iteration

Reduce the learning rate $\alpha = 0{,}5 \times 0{,}6 = 0{,}3$

The weights update equations are now:

$$w_{ij}(new) = w_{ij}(old) + 0{,}3\,[x_i - w_{ij}(old)] = 0{,}7\; w_{ij}(old) + 0{,}3\; x_i$$

The weight matrix after the 2nd epoch of training is:

$$\begin{bmatrix} 0{,}016 & 0{,}980 \\ 0{,}047 & 0{,}360 \\ 0{,}630 & 0{,}055 \\ 0{,}999 & 0{,}024 \end{bmatrix}$$

SOM-Example

(1, 1, 0, 0); (0, 0, 0, 1); (1, 0, 0, 0); (0, 0, 1, 1)

$$\begin{bmatrix} 0{,}2 & 0{,}8 \\ 0{,}6 & 0{,}4 \\ 0{,}5 & 0{,}7 \\ 0{,}9 & 0{,}3 \end{bmatrix} \quad \begin{bmatrix} 0{,}032 & 0{,}968 \\ 0{,}096 & 0{,}304 \\ 0{,}680 & 0{,}112 \\ 0{,}984 & 0{,}048 \end{bmatrix} \quad \begin{bmatrix} 1{,}5e-7 & 1{,}0000 \\ 4{,}6e-7 & 0{,}3700 \\ 0{,}6300 & 5{,}4e-7 \\ 1{,}0000 & 2{,}3e-7 \end{bmatrix} \quad \begin{bmatrix} 1{,}9e-19 & 1{,}0000 \\ 5{,}7e-15 & 0{,}4700 \\ 0{,}5300 & 6{,}6e-15 \\ 1{,}0000 & 2{,}8e-15 \end{bmatrix} \quad \begin{bmatrix} 6{,}7e-17 & 1{,}0000 \\ 2{,}0e-16 & 0{,}4900 \\ 0{,}5100 & 2{,}3e-16 \\ 1{,}0000 & 1{,}0e-16 \end{bmatrix}$$

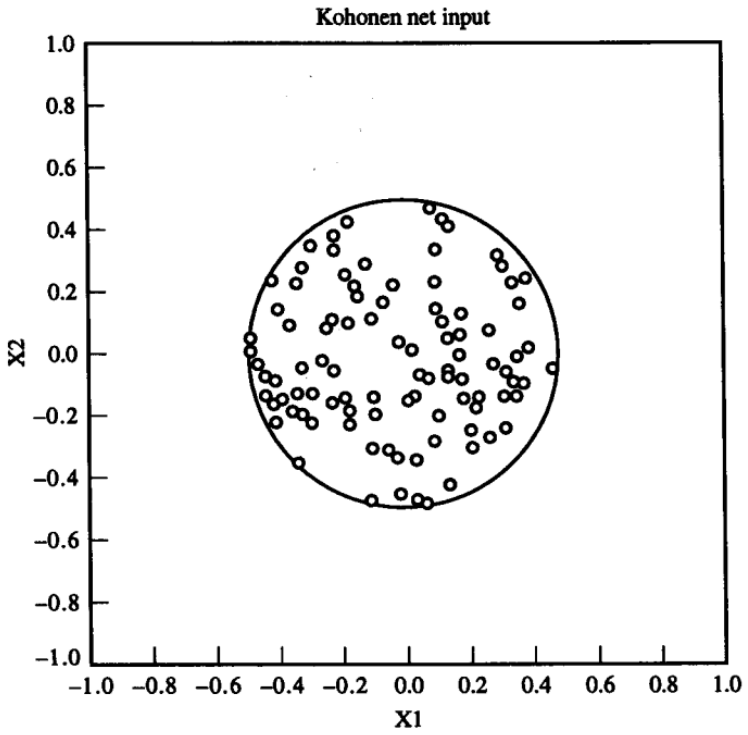
0 1 10 50 100

the average of the two vects in cluster 1

$$\begin{bmatrix} 0{,}0 & 1{,}0 \\ 0{,}0 & 0{,}5 \\ 0{,}5 & 0{,}0 \\ 1{,}0 & 0{,}0 \end{bmatrix}$$

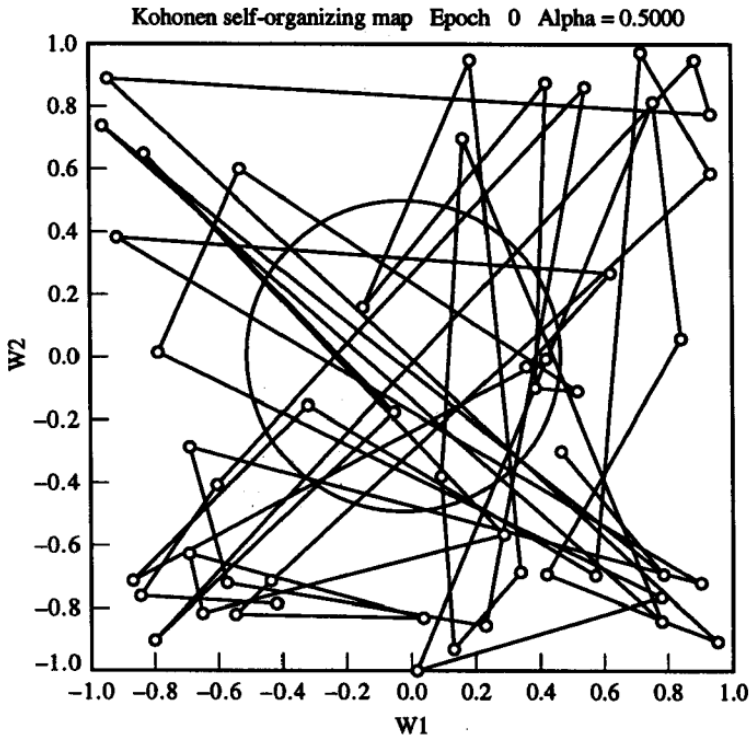
the average of the two vects in cluster 2

SOM-Example



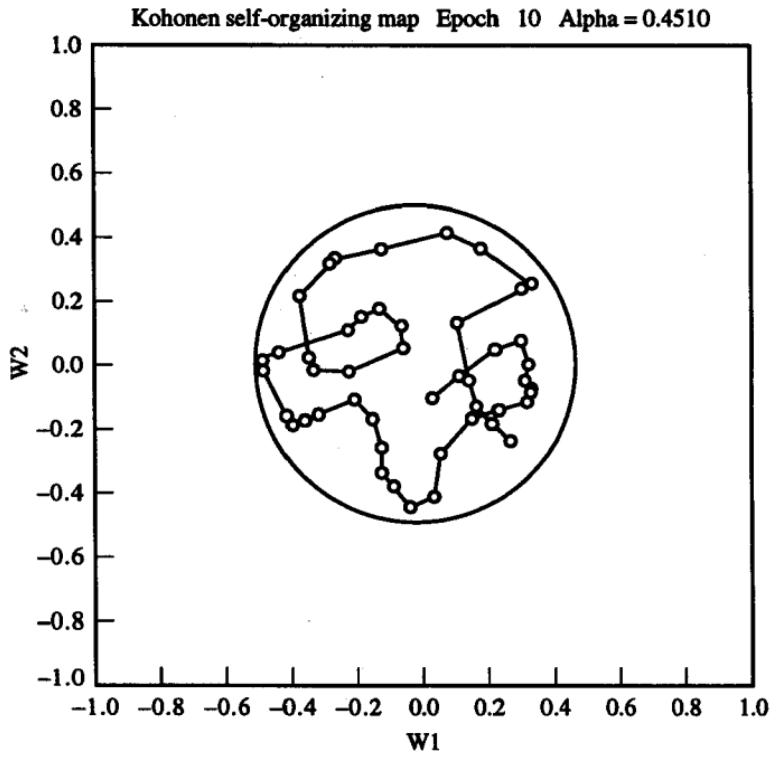
- The 100 input vectors are chosen randomly from within a circle of radius 0.5 (centered at the origin).
- we assume a linear structure.
- The initial learning rate is 0.5; it is reduced linearly to 0.01 over 100epochs
- the winning unit and its nearest neighbor unit on either side (units J, J + 1, and J - 1) are allowed to learn.

SOM-Example

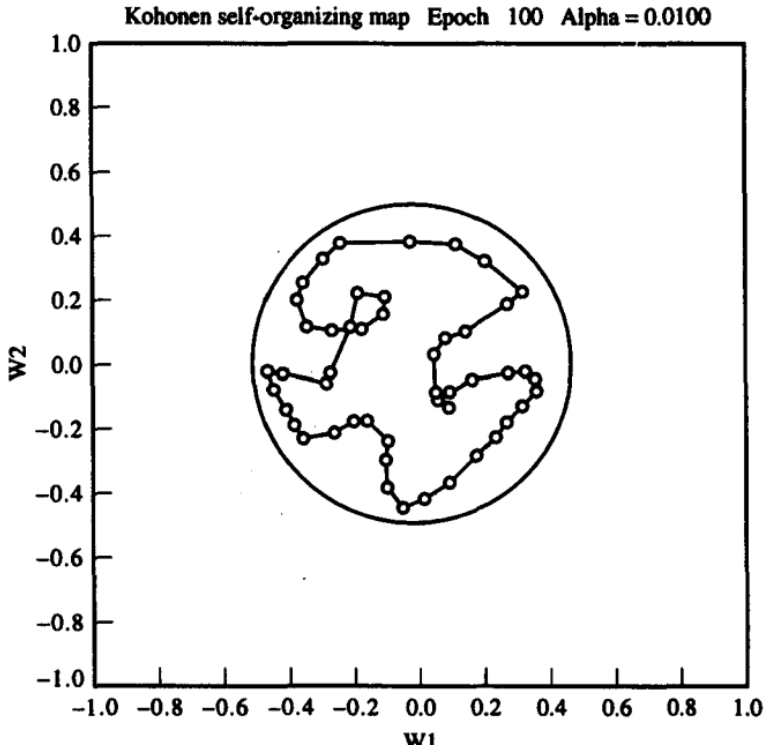


- The initial weights are chosen randomly, with each component having a value between - 1 and 1.
- There are 50 cluster units.

SOM-Example



cluster units after 10 epochs



cluster units after 100 epochs

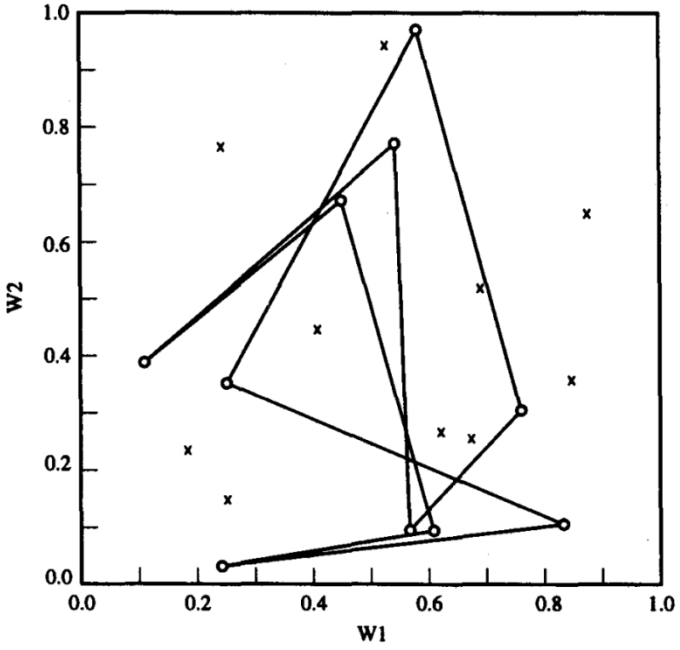
SOM-Example

- Traveling Salesman Problem(TSP)

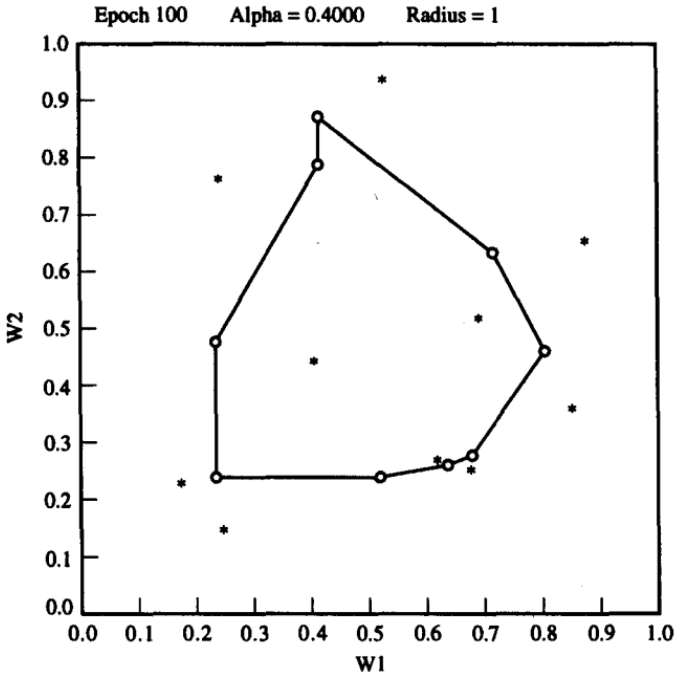
The aim of the TSP is to find a tour of a given set of cities that is of **minimum length**. A tour consists of visiting each city **exactly once** and **returning to the starting city.**

- The net uses the city coordinates as input (n=2); there are as many cluster units as there are cities to be visited. The net has a linear topology (with the first and last unit also connected).

SOM-Example

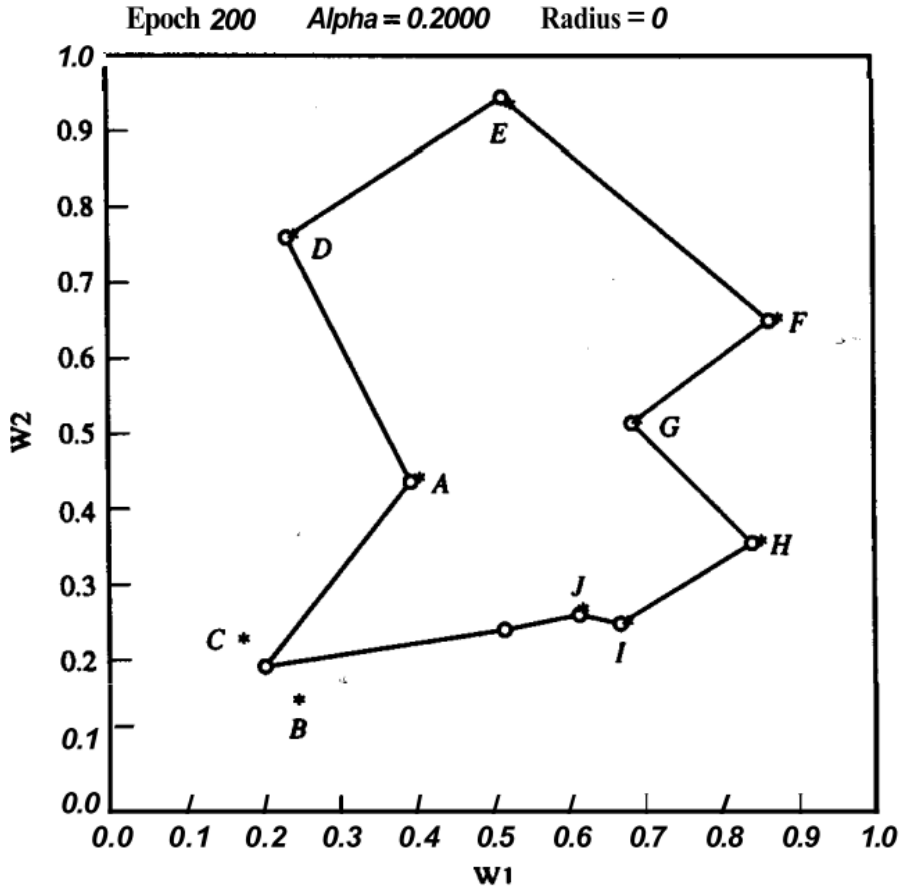


Initial position of cluster units and location of cities



the result after 100 epochs of training with R=1 (learning rate decreasing from 0.5 to 0.4)

SOM-Example

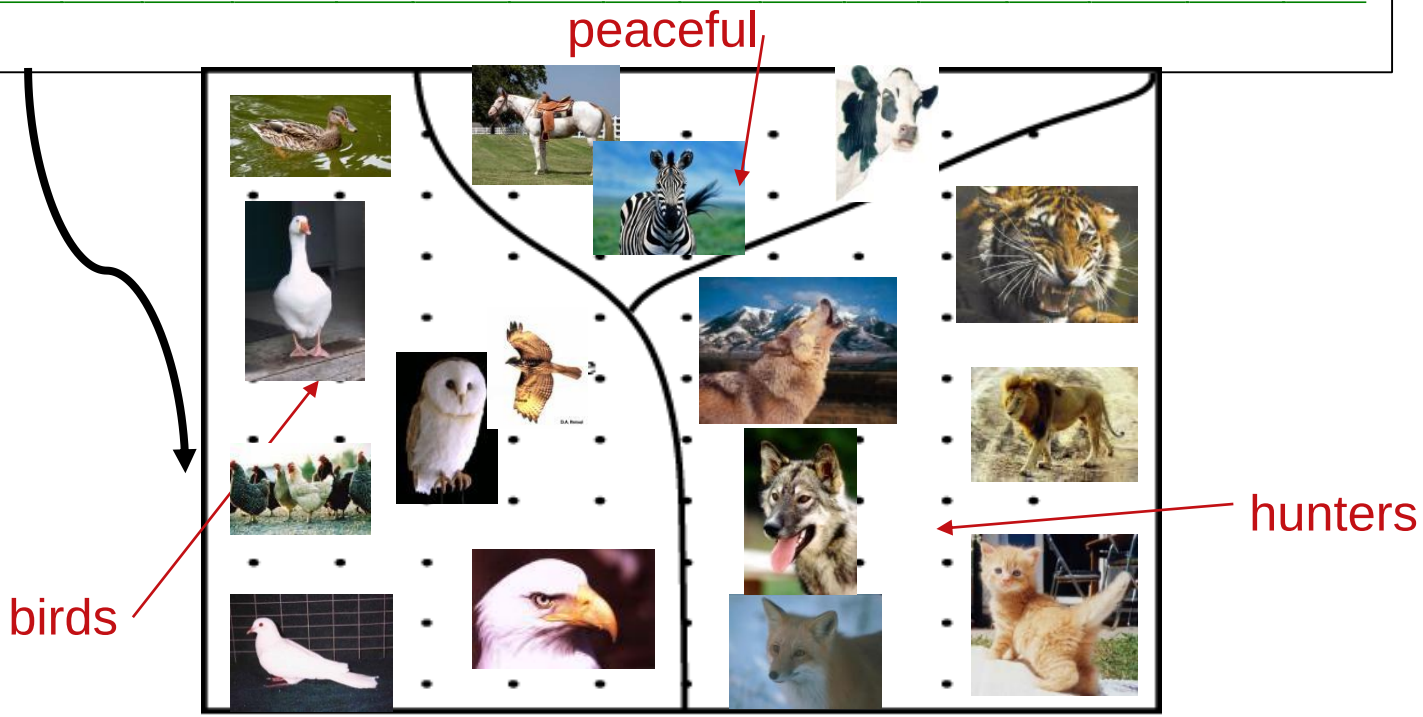


Two candidate solutions

- ADEFGHIJBC
- ADEFGHIJCB

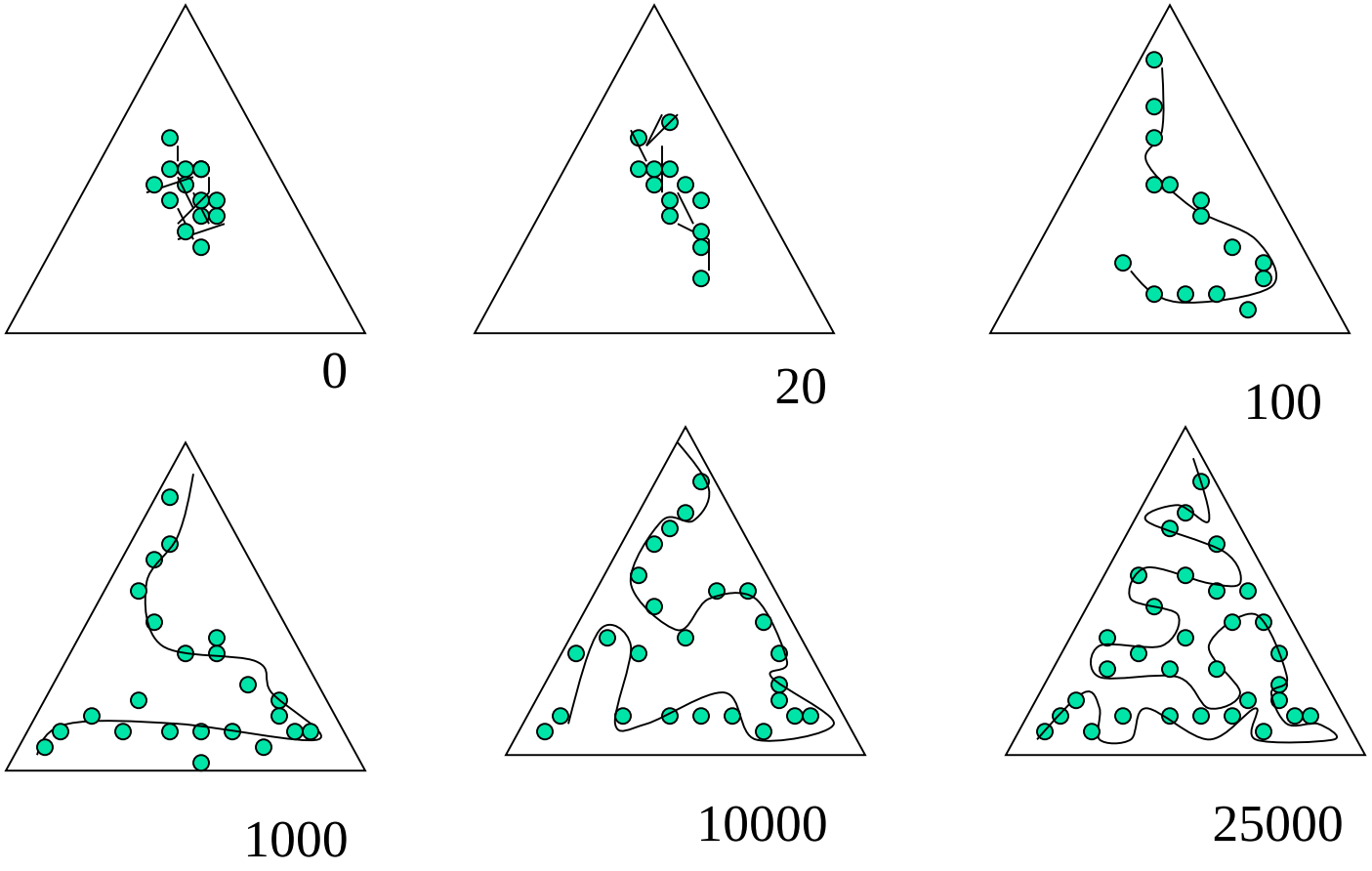
SOM Example -Clustering

		Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
is	Small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	Medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	Big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
has	2 legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	Mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	Feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
likes to	Hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	Run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	Fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	Swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

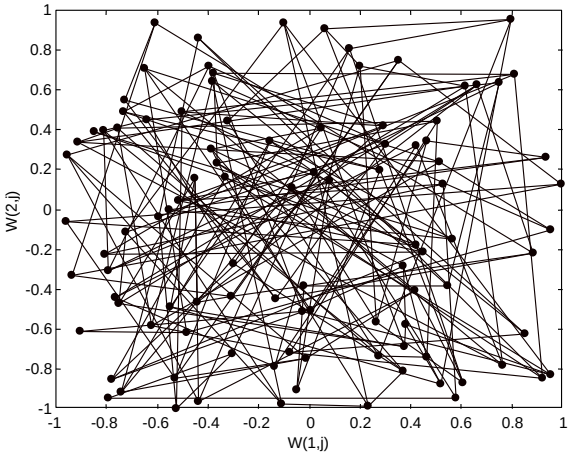


SOM-Example

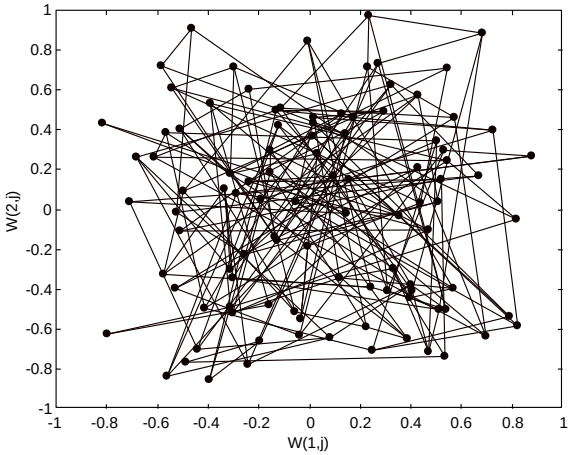
Learning a one-dimensional representation of a two-dimensional (triangular) input space



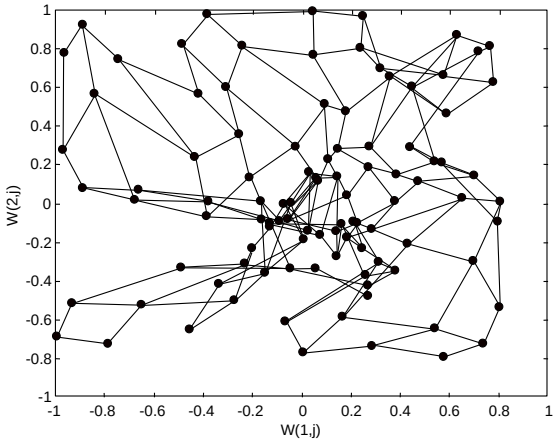
SOM-Example



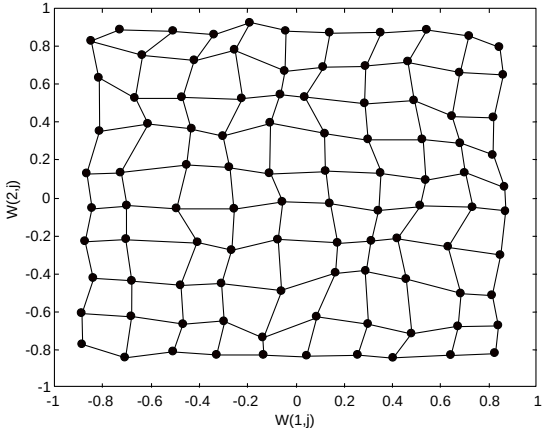
Initial random network



After 100 steps

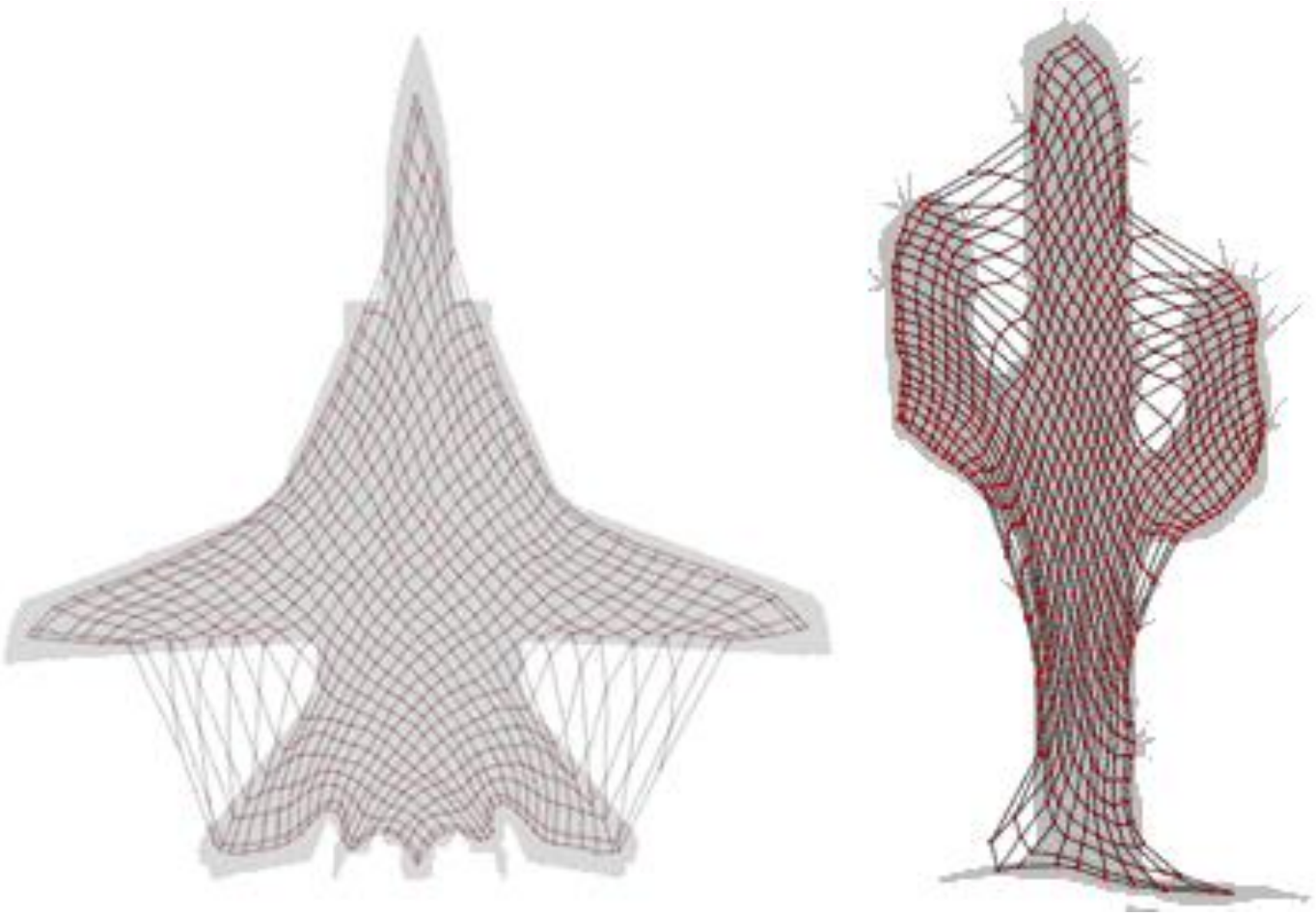


After 1000 steps



After 10000 steps

SOM-Example




Questions