International University for Science & Technology (IUST)
Department of Computer &Informatics Engineering
Neural Networks unit (5)

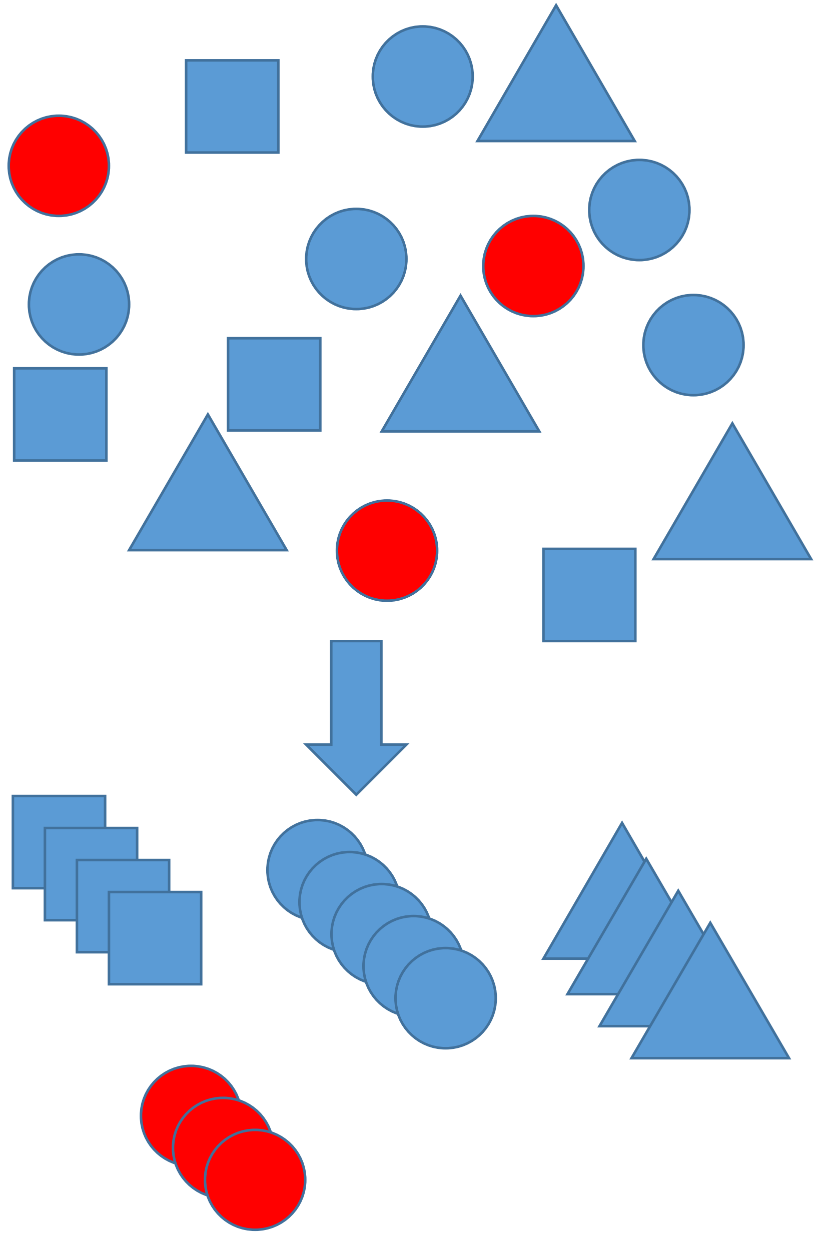
Unsupervised learning neural networks
Self organizing maps (SOM)

Neural Networks

Dr. Ali Mayya

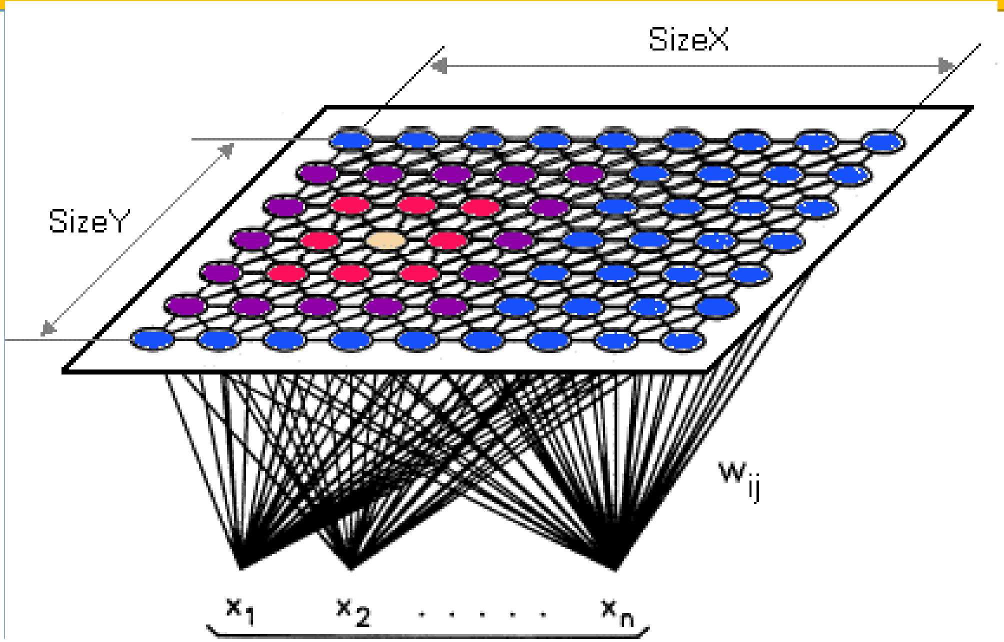
Unsupervised learning

- Self-organizing neural (SOM) group similar input vectors together **without the use of target vectors**.
- **Only input vectors** are given to the network during training.
- The learning algorithm then modifies the network weights so that ***similar input vectors*** produce the same output vector.
- ***Unsupervised training*** is good for finding hidden structure or features within a set of patterns.
- E.g. the network is allowed to cluster without being biased towards looking for a pre-defined output or classification.
- What the network learns is entirely **data driven**.
- After learning it is often necessary to examine what the clusters or output units have come to mean.

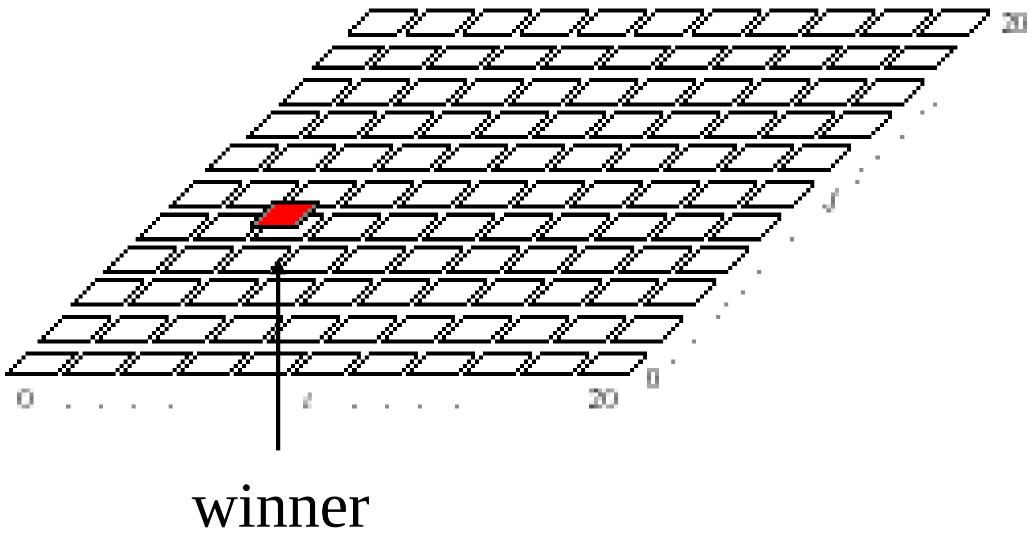


SOM

- SOM is based on the idea of unsupervised learning in which input is mapped to the ***best matching unit.***
- Maps **n-dimensional** input space onto a **two dimensional map.**
- Preserves topological relationships in the data
- SOM seeks to find the **winning neuron** of the input layer: it's the neuron whose weights closest match the inputs.



$$Y_j = \arg\min_j \left(\sum \| X_i - W_{ij} \|\right)$$



SOM- Kohonen Learning Algorithm

- KLA is the algorithm of learning unsupervised learning network (SOM) and is given in the following steps:

1. *Initialise weights (random values) and set topological neighbourhood and learning rate parameters*
2. *While stopping condition is false, do steps 3-8*
3. *For each input vector **X**, do steps 4-6*
4. *For each neuron j, compute Euclidean distance:*

$$Y_j = \arg\min_j \left(\sum \| X_i - W_{ij} \|\right)$$

5. *Find index j such that Y_J is a minimum*
6. *For all units j within a specified neighbourhood of j, and for all i:*

$$W_{ij}(new) = W_{ij}(old) + \alpha h_{ij}(X(i) - W_{ij}(old))$$

7. *Update learning rate (α)*
8. *Reduce topological neighbourhood at specified times*
9. *Test stopping condition.*

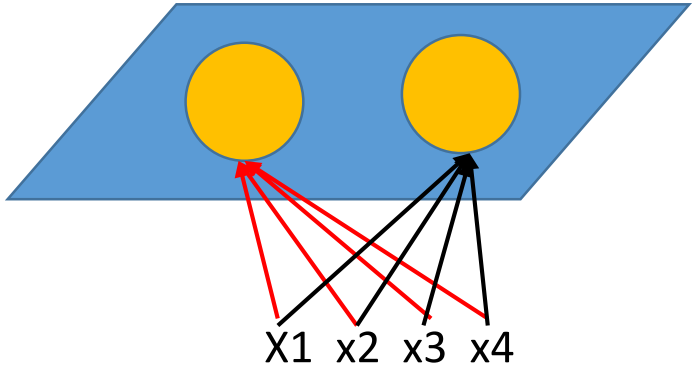
ملاحظة: يمثل التابع h تابع الجوار الطوبولوجي ويعطى الشكل الغوصي له وفق العلاقة:

$$h_{ij} = \exp(-d_{ij}^{\,2} / 2\sigma^2)$$

h determines the degree to which extent each output node receives a training adjustment from the current training pattern.

SOM- Kohonen Learning Algorithm- *Numerical Example*

- Consider two neurons (**m=2**) and the following input vectors (x1,,x2,x3,x4) = (1,1,0,0), (0,0,0,1), (1,0,0,0), (0,0,1,1)
- $W_{ij} = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{bmatrix}$
- Where j=2, i=4, α=0.6, R=0, h=1 and the decrement of learning rate
- is given as: α(t+1)=0.5α(t). Update weights (two epochs)



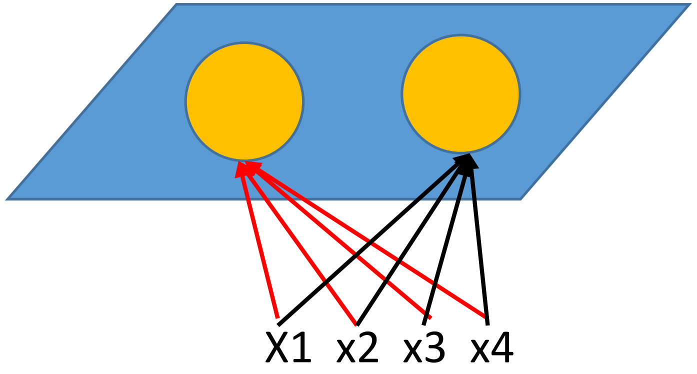
- Solution: **1.** For the first input vector (1,1,0,0) **compute distance**:
- $D(j) = \sum_j (Wi_j - xi)^2$ → Layer has two neurons → We will get two values for D
- $D(1) = (W_{11} - x_1)^2 + (W_{21} - x_2)^2 + (W_{31} - x_3)^2 + (W_{41} - x_4)^2$
- $D(1) = (0.2 - 1)^2 + (0.6 - 1)^2 + (0.5 - 0)^2 + (0.9 - 0)^2$=1.86
- $D(2) = (W_{12} - x_1)^2 + (W_{22} - x_2)^2 + (W_{32} - x_3)^2 + (W_{42} - x_4)^2$
- $D(2) = (0.8 - 1)^2 + (0.4 - 1)^2 + (0.7 - 0)^2 + (0.3 - 0)^2$=0.98

Sample1 is close to unit (neuron) 2

- **2. Define winner**: D(2)<D(1) → Input vector is closer to output node number 2 → J=2 is the winner neuron (winner unit).
- **3. Define j units** located in the winner unit's neighborhood: max(1,J-R) <= j <= min(J+R,m)
- Max(1,2-0) <= j <= min(2+0,2) → 2<=j<=2 → j=2

SOM- Kohonen Learning Algorithm- Numerical Example

- Consider two neurons (**m=2**) and the following input vectors (x1,,x2,x3,x4) = (1,1,0,0), (0,0,0,1), (1,0,0,0), (0,0,1,1)
- $W_{ij} = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{bmatrix}$
- Where j=2, i=4, α=0.6, R=0 and the decrement of learning rate
- is given as: α(t+1)=0.5α(t). Update weights (two epochs)



- **4. Weights updates:** For all units j located in the neighborhood of J and for all i, update weights:

$$W_{ij}(new) = W_{ij}(old) + \alpha(X(i) - W_{ij}(old))$$

$$W_{i2}(new) = W_{i2}(old) + \alpha(X(i) - W_{i2}(old))$$

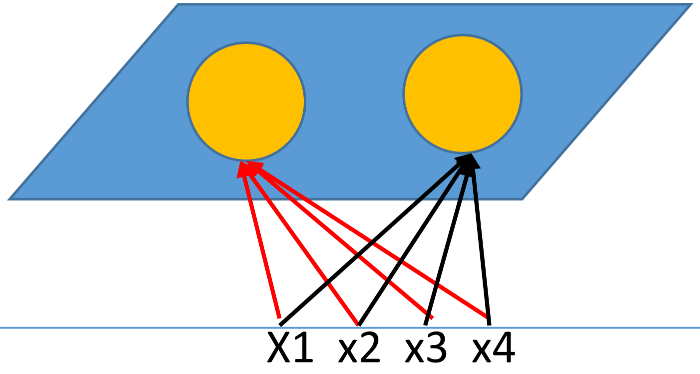
$$W_{i2}(new) = W_{i2}(old) + 0.6(X(i) - W_{i2}(old))$$

$$W_{i2}(new) = 0.4W_{i2}(old) + 0.6X(i)$$

$$= 0.4 * \begin{bmatrix} 0.8 \\ 0.4 \\ 0.7 \\ 0.3 \end{bmatrix} + 0.6 * \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.92 \\ 0.76 \\ 0.28 \\ 0.12 \end{bmatrix} \rightarrow \text{new weight matrix is: } W_{ij} = \begin{bmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{bmatrix}$$

SOM- Kohonen Learning Algorithm- *Numerical Example*

- (**m=2**) and the following input vectors (x1,,x2,x3,x4) = (1,1,0,0), (0,0,0,1), (1,0,0,0), (0,0,1,1)
- $new\ W_{ij} = \begin{bmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{bmatrix}$
- Where j=2, i=4, α=0.6, R=0 and the decrement of learning rate
- is given as: α(t+1)=0.5α(t).



- **5. Repeat** for the **second** input vector: (0,0,0,1) **<u>compute distance</u>**:
- $D(1) = (0.2-0)^2+(0.6-0)^2+(0.5-0)^2+(0.9-1)^2$=0.66
- $D(2) = (0.8-0)^2+(0.4-0)^2+(0.7-0)^2+(0.3-1)^2$=2.2768
- **<u>Define winner</u>**: D(1)<D(2) → Input vector is closer to output node number 1 → J=1 is the winner neuron (winner unit).
- **<u>Define j units</u>** located in the winner unit's neighborhood: max(1,J-R) <= j <= min(J+R,m)
- Max(1,1-0) <= j <= min(1+0,2) → 1<=j<=1 → j=1
- Update weights:

Sample2 is close to unit (neuron) 1

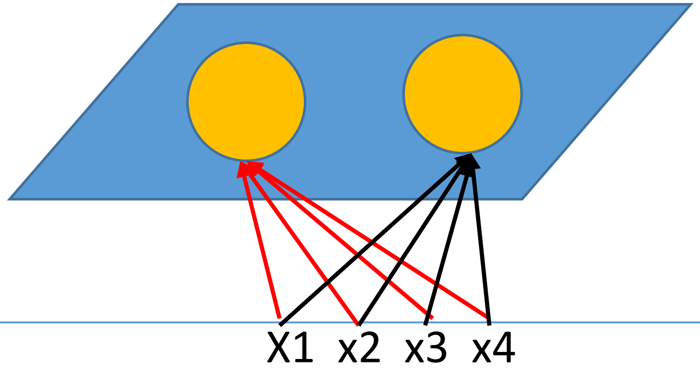
$$W_{i1}(new) = W_{i1}(old) + 0.6(X(i) - W_{i1}(old))$$

$$W_{i1}(new) = 0.4W_{i1}(old) + 0.6X(i) \quad = 0.4 * \begin{bmatrix} 0.2 \\ 0.6 \\ 0.5 \\ 0.9 \end{bmatrix} + 0.6* \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.08 \\ 0.24 \\ 0.2 \\ 0.96 \end{bmatrix}$$

→ new weight matrix is: $W_{ij} = \begin{bmatrix} 0.08 & 0.92 \\ 0.24 & 0.76 \\ 0.2 & 0.28 \\ 0.96 & 0.12 \end{bmatrix}$

SOM- Kohonen Learning Algorithm- *Numerical Example*

- (**m=2**) and the following input vectors (x1,,x2,x3,x4) = (1,1,0,0), (0,0,0,1), (1,0,0,0), (0,0,1,1)
- New $W_{ij} = \begin{bmatrix} 0.08 & 0.92 \\ 0.24 & 0.76 \\ 0.2 & 0.28 \\ 0.96 & 0.12 \end{bmatrix}$
- Where j=2, i=4, α=0.6, R=0 and the decrement of learning rate
- is given as: α(t+1)=0.5α(t).



- **6. Repeat** for the **third** input vector: (1,0,0,0) **compute distance**:
- $D(1) = (0.2-1)^2 + (0.6-0)^2 + (0.5-0)^2 + (0.9-0)^2 = 1.8656$
- $D(2) = (0.8-1)^2 + (0.4-0)^2 + (0.7-0)^2 + (0.3-0)^2 = 0.6768$
- **Define winner**: D(2)<D(1) → Input vector is closer to output node number 1 → J=2 is the winner neuron (winner unit).
- **Define j units** located in the winner unit's neighborhood: max(1,J-R) <= j <= min(J+R,m)
- Max(1,2-0) <= j <= min(2+0,2) → 2<=j<=2 → j=2
- Update weights:

Sample3 is close to unit (neuron) 2

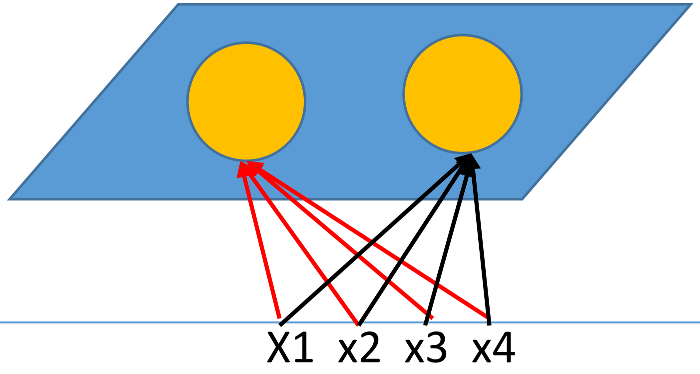
$$W_{i2}(new) = W_{i2}(old) + 0.6(X(i) - W_{i2}(old))$$

$$W_{i2}(new) = 0.4W_{i2}(old) + 0.6X(i) = 0.4 * \begin{bmatrix} 0.92 \\ 0.76 \\ 0.28 \\ 0.12 \end{bmatrix} + 0.6 * \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.968 \\ 0.304 \\ 0.112 \\ 0.048 \end{bmatrix}$$

→new weight matrix is: $W_{ij} = \begin{bmatrix} 0.08 & 0.968 \\ 0.24 & 0.304 \\ 0.2 & 0.112 \\ 0.96 & 0.048 \end{bmatrix}$

SOM- Kohonen Learning Algorithm- *Numerical Example*

- (**m=2**) and the following input vectors (x1,,x2,x3,x4) = (1,1,0,0), (0,0,0,1), (1,0,0,0), (0,0,1,1)
- New $W_{ij} = \begin{bmatrix} 0.08 & 0.968 \\ 0.24 & 0.304 \\ 0.2 & 0.112 \\ 0.96 & 0.048 \end{bmatrix}$
- Where j=2, i=4, α=0.6, R=0 and the decrement of learning rate
- is given as: α(t+1)=0.5α(t).



- **7. Repeat** for the **fourth** input vector: (0,0,1,1) **compute distance**:
- $D(1) = (0.2-0)^2+(0.6-0)^2+(0.5-1)^2+(0.9-1)^2$=0.7056
- $D(2) = (0.8-0)^2+(0.4-0)^2+(0.7-1)^2+(0.3-1)^2$=2.724
- **Define winner**: D(1)<D(2) → Input vector is closer to output node number 1 → J=1 is the winner neuron (winner unit).
- **Define j units** located in the winner unit's neighborhood: max(1,J-R) <= j <= min(J+R,m)
- Max(1,1-0) <= j <= min(1+0,2) → 1<=j<=1 → j=1
- Update weights:

Sample4 is close to unit (neuron) 1

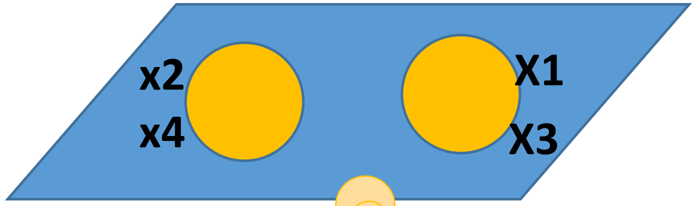
$$W_{i1}(new) = W_{i1}(old) + 0.6(X(i) - W_{i1}(old))$$

$$W_{i1}(new) = 0.4W_{i1}(old) + 0.6X(i) \quad = 0.4 * \begin{bmatrix} 0.08 \\ 0.24 \\ 0.2 \\ 0.96 \end{bmatrix} + 0.6 * \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.032 \\ 0.096 \\ 0.68 \\ 0.984 \end{bmatrix}$$

→new weight matrix is: $W_{ij} = \begin{bmatrix} 0.032 & 0.968 \\ 0.096 & 0.304 \\ 0.68 & 0.112 \\ 0.984 & 0.048 \end{bmatrix}$

SOM- Kohonen Learning Algorithm- *Numerical Example*

- (**m=2**) and the following input vectors (x1,,x2,x3,x4) = (1,1,0,0), (0,0,0,1), (1,0,0,0), (0,0,1,1)
- New $W_{ij} = \begin{bmatrix} 0.032 & 0.968 \\ 0.096 & 0.304 \\ 0.68 & 0.112 \\ 0.984 & 0.048 \end{bmatrix}$
- Where j=2, i=4, α=0.6, R=0 and the decrement of learning rate
- is given as: α(t+1)=0.5α(t).



At the end of epoch1, x2 and x4 are clustered together (unit1), while x1 and x3 are clustered together (unit 1)

- **The first training epoch has ended!**
- **8. Decrease learning rate:**
- α(t+1)=0.5α(t) = 0.5*0.6 = 0.3 → Repeat same steps of the first epoch taking into consideration that the weight update equation is now:

$$W_{ij}(new) = W_{ij}(old) + 0.6(X(i) - W_{ij}(old))$$

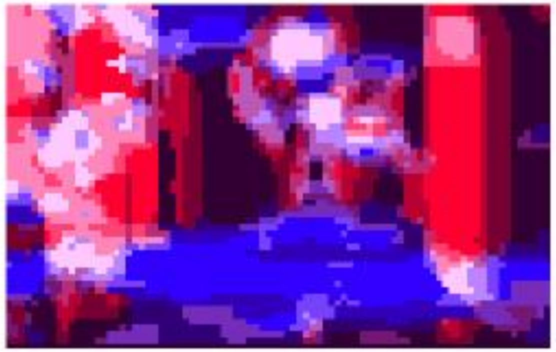
$$W_{ij}(new) = 0.7W_{i1}(old) + 0.3X(i)$$

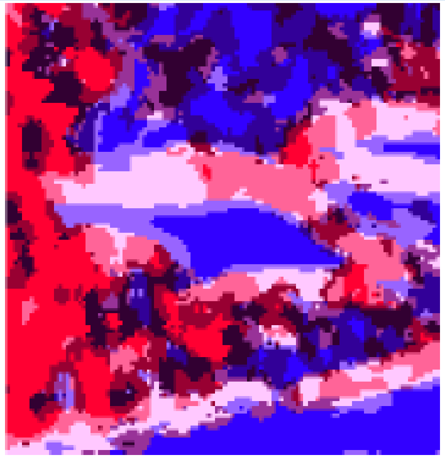
$$W_{ij} of\ epoch\ 2 = \begin{bmatrix} 0.0157 & 0.9843 \\ 0.047 & 0.359 \\ 0.6332 & 0.0549 \\ 0.9922 & 0.0235 \end{bmatrix}$$

$$W_{ij} of\ final\ epoch\ 55 = \begin{bmatrix} 0 & 1 \\ 0 & 0.5 \\ 0.5 & 0 \\ 1 & 0 \end{bmatrix}$$

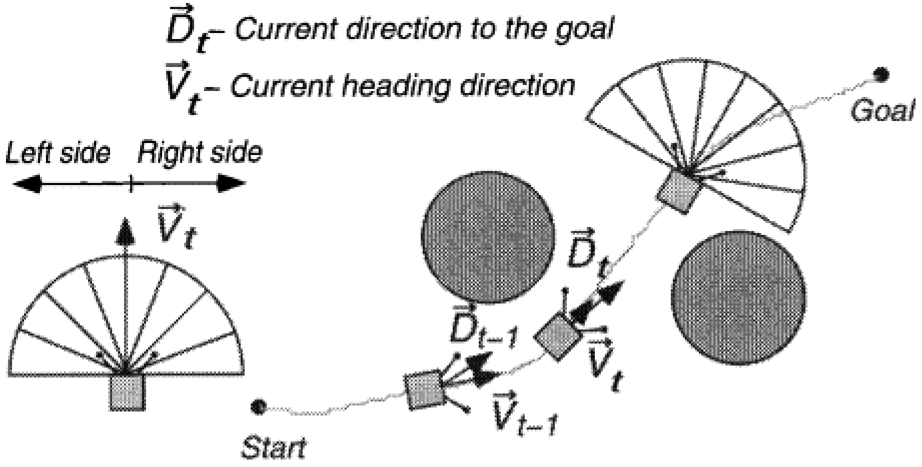
SOM- Applications

- Image Segmentation





- Vision based autonomous robot navigation systems

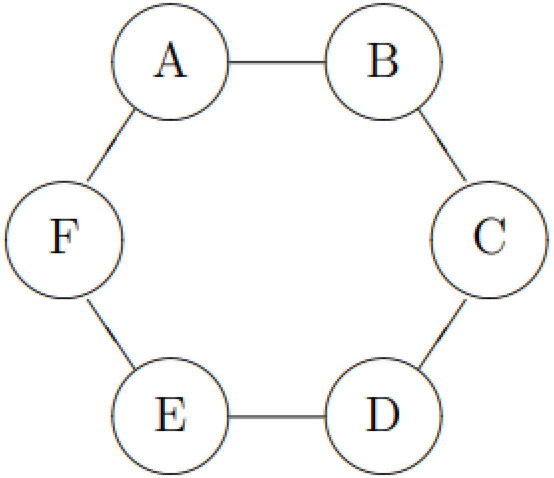


https://www.sciencedirect.com/science/article/pii/B9780080925097500142

SOM- Example2 to be solved by students

- Consider the following self–organising map:
- Each of the output nodes has two inputs x_1 and x_2 (not shown on the diagram). Thus, each node has two weights corresponding to these inputs: w_1 and w_2. The values of the weights for all output in the SOM nodes are given in the table below:

Node	A	B	C	D	E	F
w_1	-1	0	3	-2	3	4
w_2	2	4	-2	-3	2	-1



- For an input pattern $x = (x_1, x_2)$ the winner is determined using Euclidean distance Calculate which of the six output nodes is the winner if the input pattern is X=(2,-4)

SOM- Example2 to be solved by students

- Solution:

$$
\begin{aligned}
d(\mathbf{x}, \mathbf{w}_A) &= \sqrt{|2+1|^2 + |-4-2|^2} = \sqrt{9+36} = \sqrt{45} \\
d(\mathbf{x}, \mathbf{w}_B) &= \sqrt{|2-0|^2 + |-4-4|^2} = \sqrt{4+64} = \sqrt{68} \\
d(\mathbf{x}, \mathbf{w}_C) &= \sqrt{|2-3|^2 + |-4+2|^2} = \sqrt{1+4} = \sqrt{5} \\
d(\mathbf{x}, \mathbf{w}_D) &= \sqrt{|2+2|^2 + |-4+3|^2} = \sqrt{16+1} = \sqrt{17} \\
d(\mathbf{x}, \mathbf{w}_E) &= \sqrt{|2-3|^2 + |-4-2|^2} = \sqrt{1+36} = \sqrt{37} \\
d(\mathbf{x}, \mathbf{w}_F) &= \sqrt{|2-4|^2 + |-4+1|^2} = \sqrt{4+9} = \sqrt{13}
\end{aligned}
$$

The winner is the node with the smallest distance from x. Thus, in this case the **winner** is **node C** (because √ 5 is the smallest distance here).

SOM- Example2-B to be solved by students

- After the winner for a given input x has been identified, the weights of the nodes in SOM are adjusted using adaptation formula:

$$\mathbf{w}' = \mathbf{w} + \alpha\, h[\mathbf{x} - \mathbf{w}]$$

- where w_0 is the new weight vector, α is the learning rate, h is the neighborhood function. Let α = 0.5 and the neighborhood be defined as:

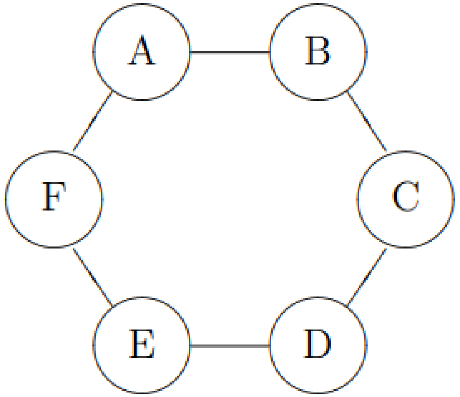
$$h = \begin{cases} 1 & \text{if the node is the winner} \\ 0.5 & \text{if the node is immediate neighbour of the winner} \\ 0 & \text{otherwise} \end{cases}$$

- Adjust the weights in the SOM.

Node	A	B	C	D	E	F
w_1	-1	0	3	-2	3	4
w_2	2	4	-2	-3	2	-1

SOM- Example2-B to be solved by students

- Example 2-B solution: since X=(2,-4), and w1,w2 for C node is (3,-2) which is the winner neuron →



$$\mathbf{x}-\mathbf{w}_C=\begin{pmatrix}2\\-4\end{pmatrix}-\begin{pmatrix}3\\-2\end{pmatrix}=\begin{pmatrix}-1\\-2\end{pmatrix}$$

$$\alpha\, h[\mathbf{x}-\mathbf{w}_C]=0.5\cdot 1\cdot\begin{pmatrix}-1\\-2\end{pmatrix}=\begin{pmatrix}-0.5\\-1\end{pmatrix}$$

$$\mathbf{w}'_C=\mathbf{w}_C+\alpha\, h[\mathbf{x}-\mathbf{w}_C]=\begin{pmatrix}3\\-2\end{pmatrix}+\begin{pmatrix}-0.5\\-1\end{pmatrix}=\begin{pmatrix}2.5\\-3\end{pmatrix}$$

Node	A	B	C	D	E	F
w_1	-1	0	3	-2	3	4
w_2	2	4	-2	-3	2	-1

- The immediate neighbors of node C are nodes B and D. The neighborhood h = 0.5. The new weights of node B and D are:

$$\mathbf{w}'_B=\begin{pmatrix}0\\4\end{pmatrix}+0.5\cdot 0.5\cdot\left[\begin{pmatrix}2\\-4\end{pmatrix}-\begin{pmatrix}0\\4\end{pmatrix}\right]=\begin{pmatrix}0.5\\2\end{pmatrix}$$

$$\mathbf{w}'_D=\begin{pmatrix}-2\\-3\end{pmatrix}+0.5\cdot 0.5\cdot\left[\begin{pmatrix}2\\-4\end{pmatrix}-\begin{pmatrix}-2\\-3\end{pmatrix}\right]=\begin{pmatrix}-1\\-3.25\end{pmatrix}$$

Rest of neurons (A,F,E) are not immediate neighbors of C so h=0 and their weights are not updated