SELF-ORGANIZING MAP (SOM)

Kohonen Network

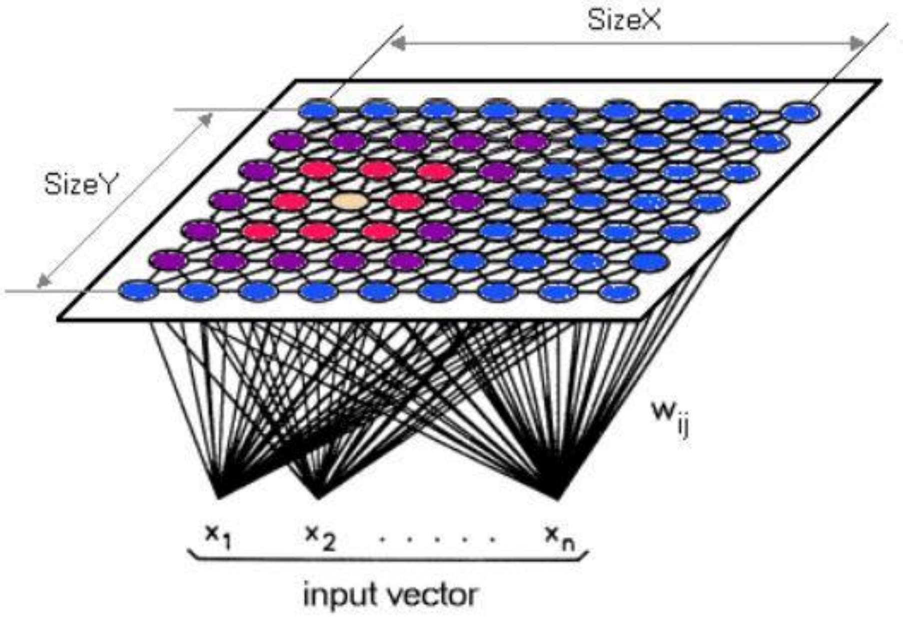
Kohonen Network

- Its un-supervised neural network (**clustering**)
- Un-supervised NN are trained by letting the network **continually adjust itself to new inputs**. They find relationships within data and can automatically define classification schemes.

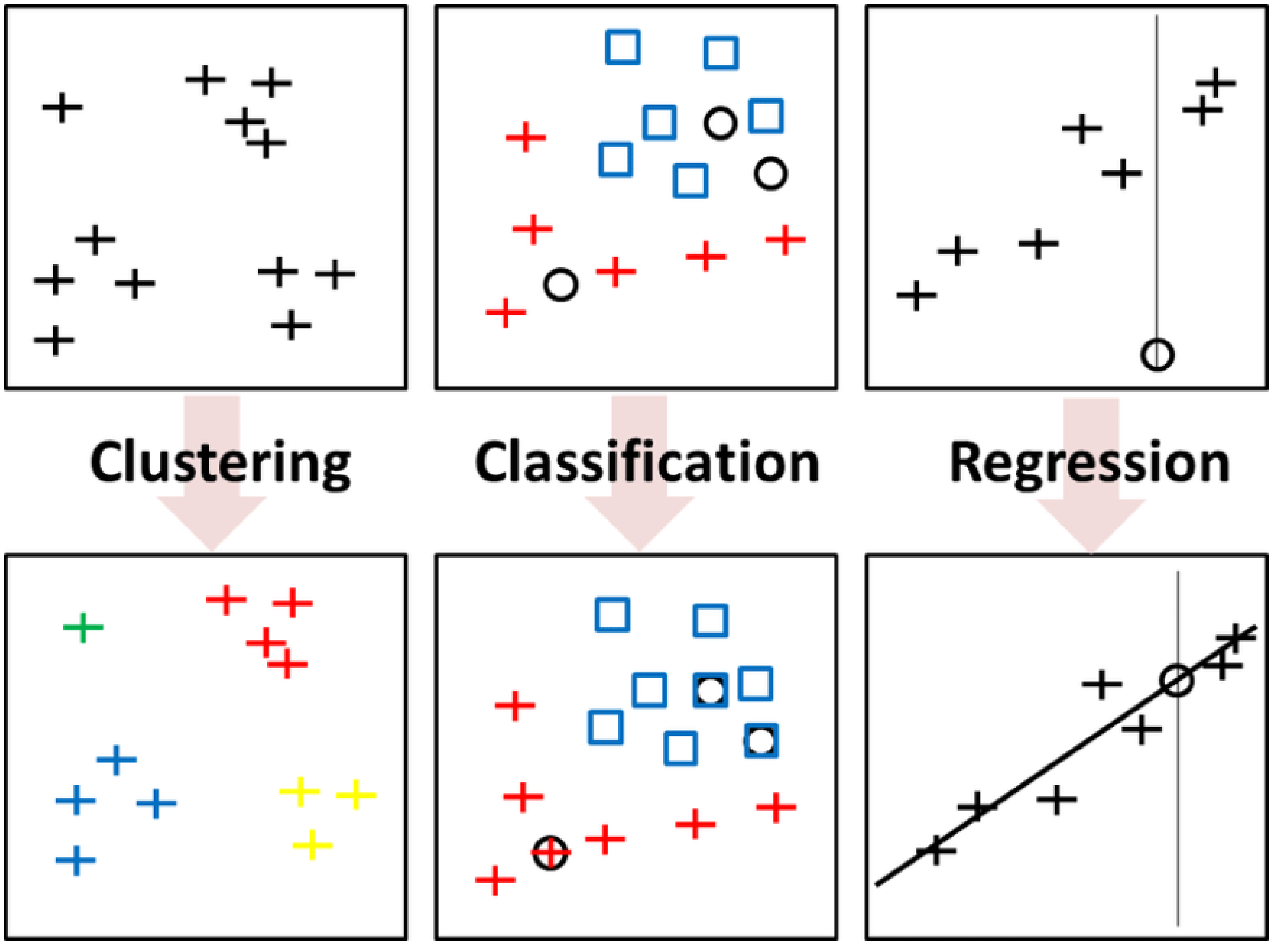
- Self -organizing maps (SOM) is a type of artificial neural network which can represent a one-dimensional input space in a two-dimensional map

Network Structure

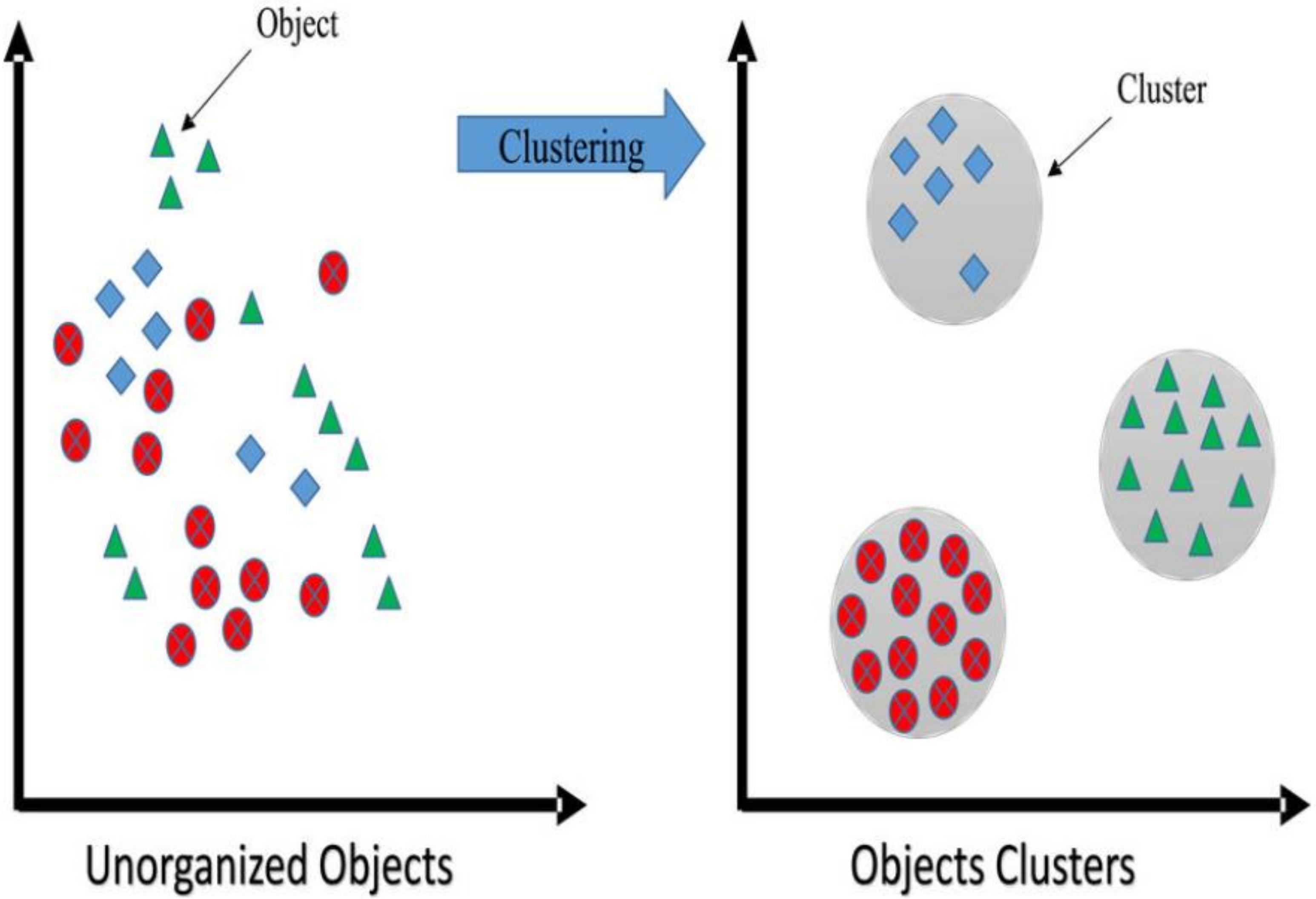
- **It has two layer:**
  - Input layer
  - Computational layer (Grid)
    - Each neuron is fully connected to all the source nodes in the input layer



Classification vs Regression vs Clustering



Clustering


Object
Clustering
Cluster
Unorganized Objects
Objects Clusters

How the algorithm learn?

- It learn to classify input vectors according to similarity. They are used for classification and pattern recognition tasks.
- The stages of the SOM algorithm can be summarized as follows:

1. **Sampling**: get a sample training input vector from the input space.
2. **Initialization**: build a grid and initialize each node with random features
3. **Matching:**
   - Compare each input with all nodes in the grid
   - Select the best matching unit (BMU)
4. **Updating**: update each node in the range of the BMU
5. **Continuation:** keep returning to step 3 until the feature map stops changing

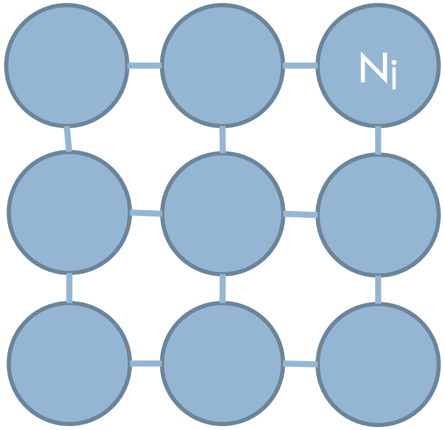
How the algorithm learn?

<u>1) Sampling</u>: get a sample training input vector from the input space.

$X=[\mathbf{X}_0,X_1,X_2,.....,X_n]$ each input is vector

$\mathbf{X}_0=[X_{00},X_{01},....X_{0m}]$

<u>2) Initialization</u>: build a grid and initialize each node with random features

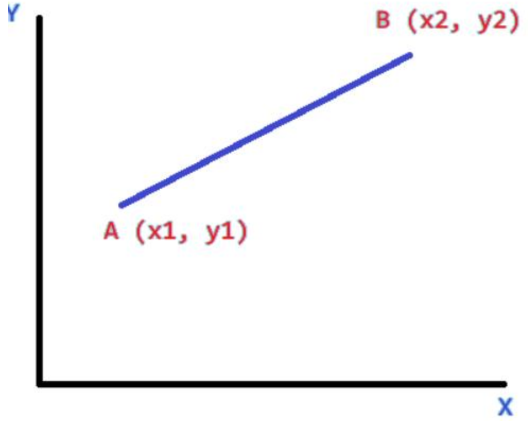


$N_j=[N_{j0}, N_{j1},...,N_{jm}]$

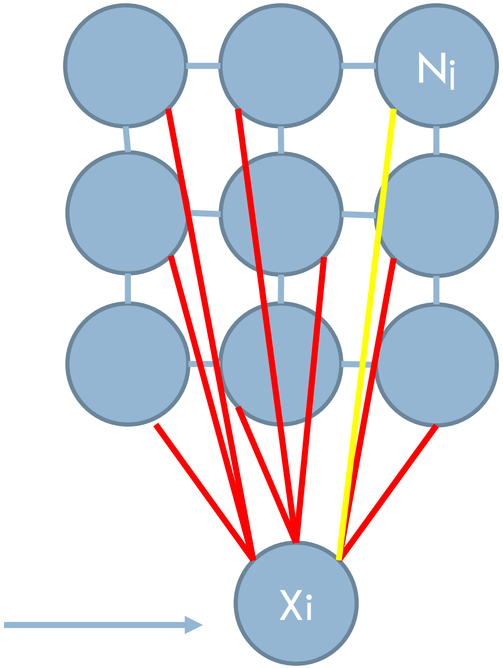
How the algorithm learn?

3) Matching:

- Compare each input with all nodes in the grid
- Select the best matching unit (BMU)



$$AB = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



N_j=[N_{j0}, N_{j1},…,N_{jm}]

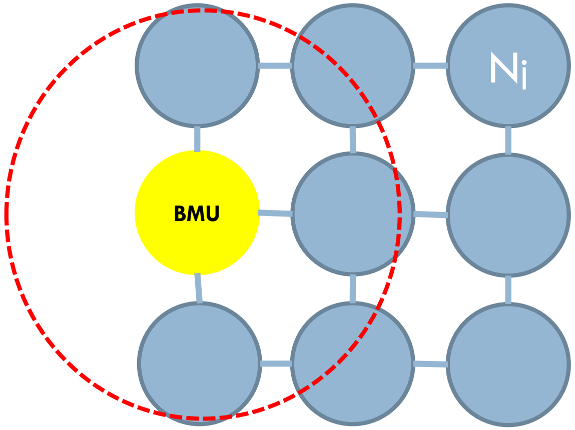
$$D_{ij} = \sqrt{\sum_{k=1}^{m} (x_{ik} - N_{jk})^2}$$

X_0,X_1,X_2…,X_n

X_i=[X_{i0}, X_{i1},…,X_{im}]

How the algorithm learn?

<u>4) Updating</u>: update each node in the range of the BMU



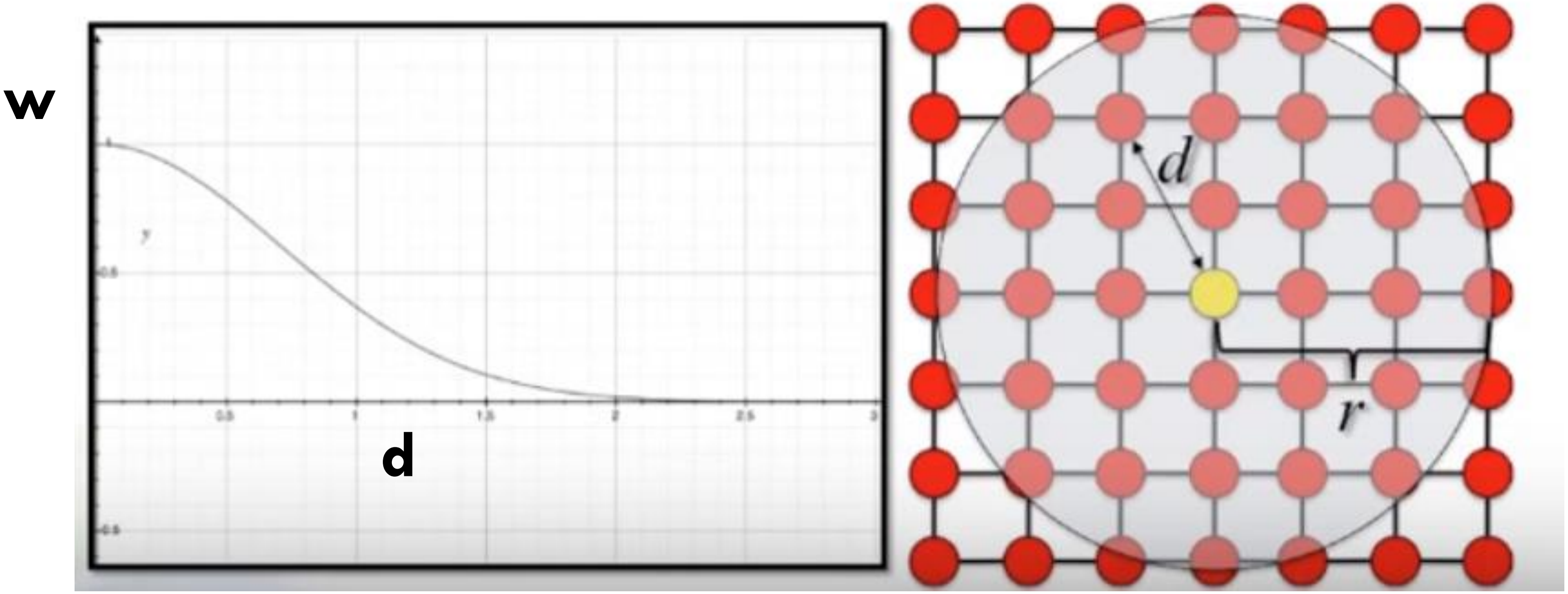
$$N_j = N_j + \eta w_j \left(x_i - N_j\right)$$

Where w_j is a weight parameter that depends on the distance between the node and the BMU and η is the learning rate

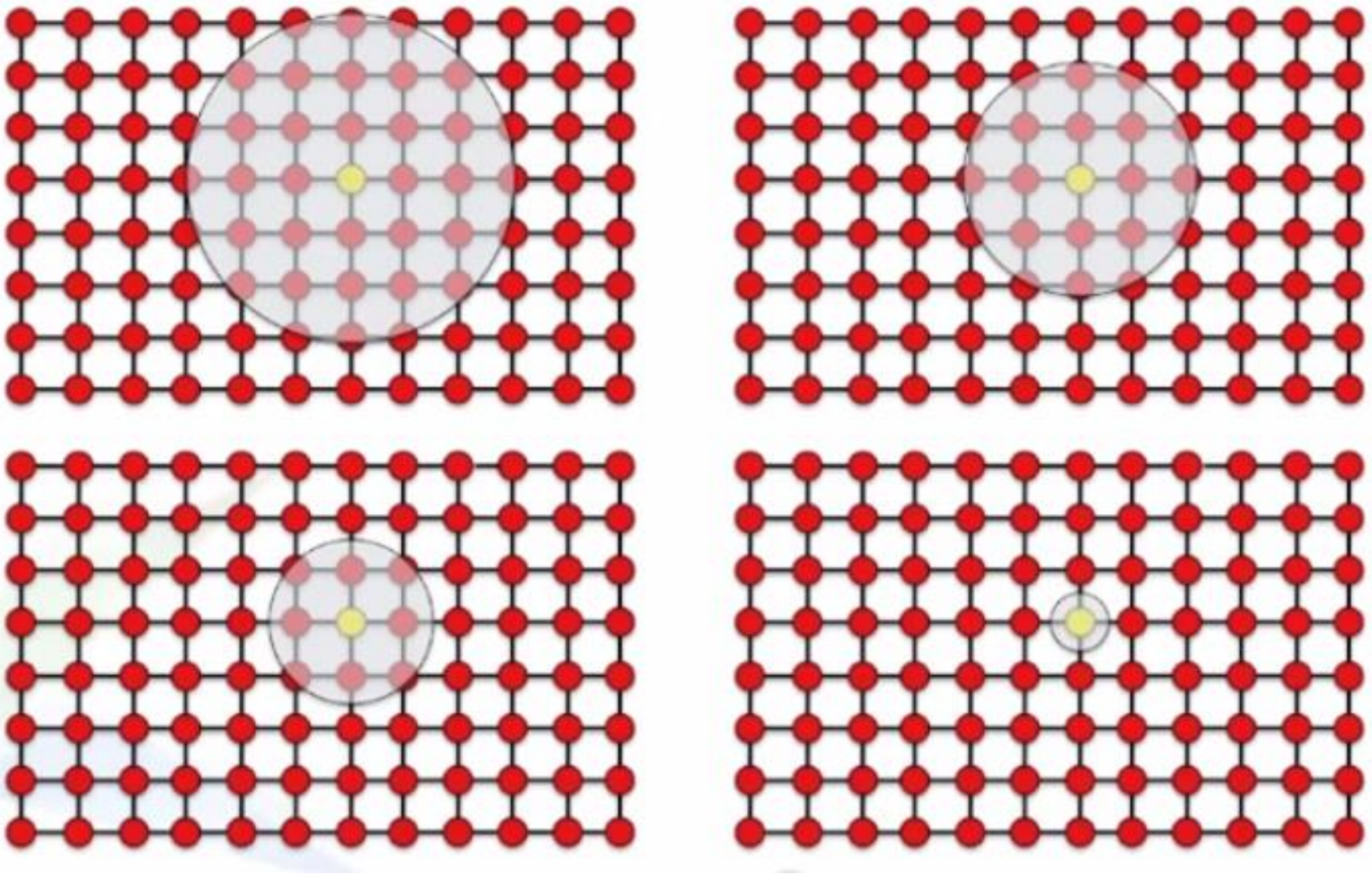
<u>5) Continuation:</u> keep returning to step 3 until the feature map stops changing

- Repeat with different input
  - Repeat with smaller radius (Go again with the input but with smaller radius)

$$w_j = e^{-\frac{d_j^2}{2r^2}}$$



Repeat with smaller radius

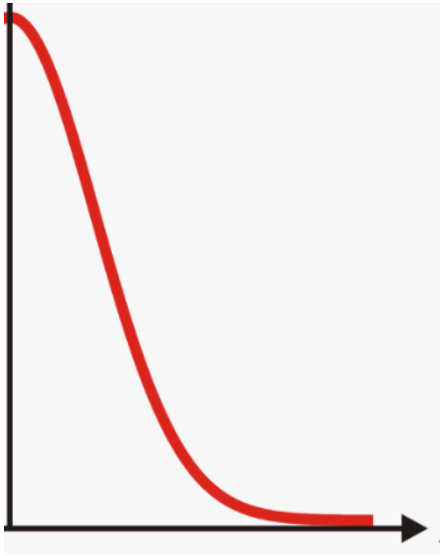


How to calculate radius

□ **Radius**

$$r(T) = r(0) * e^{\left(-t\left(\frac{\ln(Grid\ size)}{Max\ t}\right)\right)}$$

Grid Size

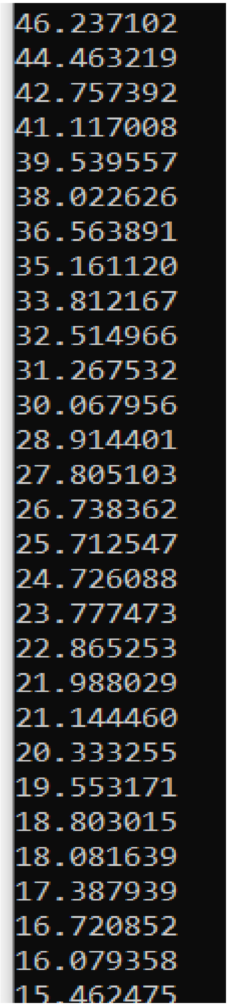


Initial

r=50*exp(0)=50

Last iteration

r=50*1/50=1

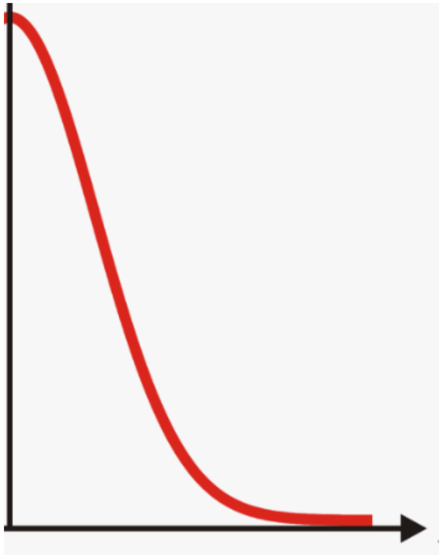


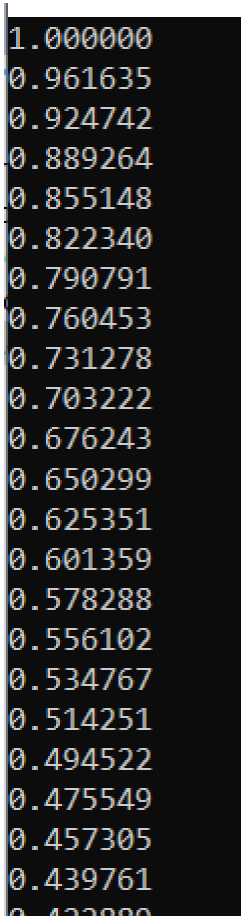
```
46.237102
44.463219
42.757392
41.117008
39.539557
38.022626
36.563891
35.161120
33.812167
32.514966
31.267532
30.067956
28.914401
27.805103
26.738362
25.712547
24.726088
23.777473
22.865253
21.988029
21.144460
20.333255
19.553171
18.803015
18.081639
17.387939
16.720852
16.079358
15.462475
```

How to calculate learning rate

- **Learning Rate**

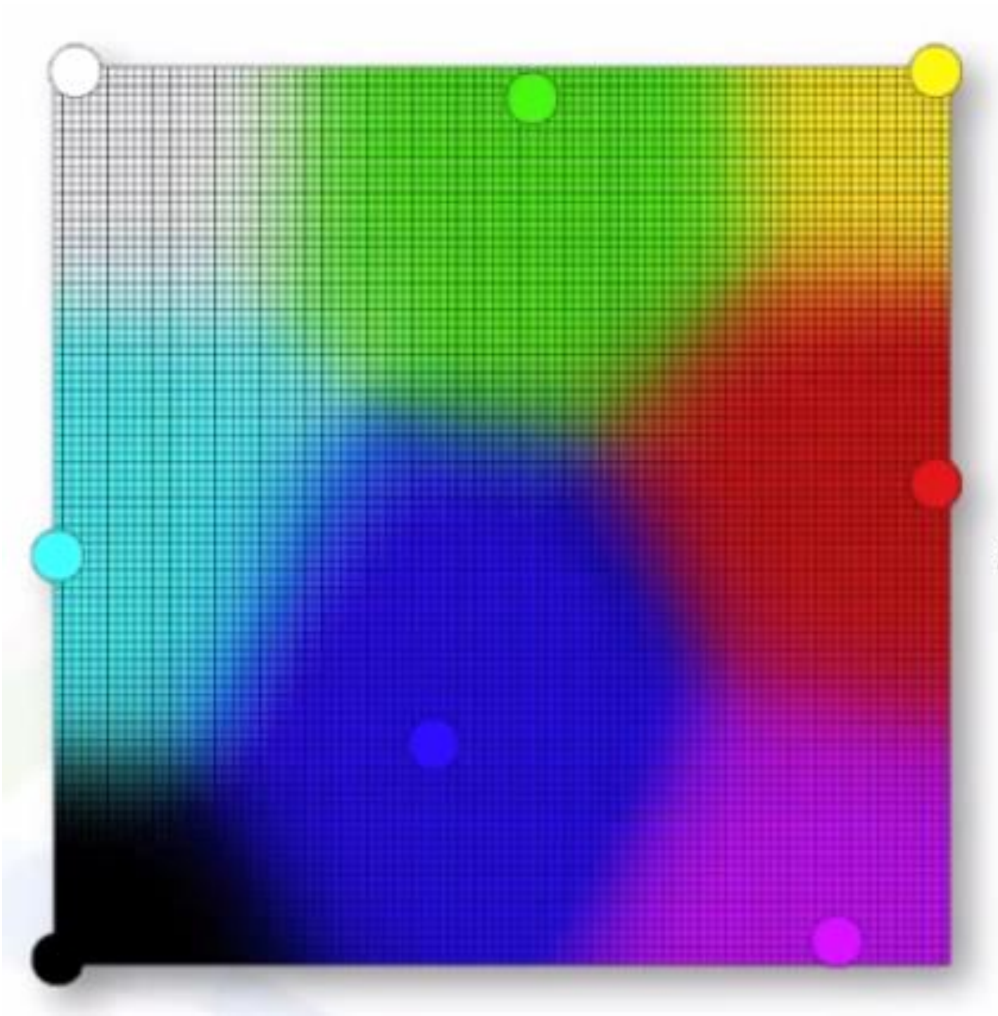
$$\alpha(T) = \alpha(0) * e^{\left(-t\left(\frac{\ln\left(Grid\ size\right)}{Max\ t}\right)\right)}$$





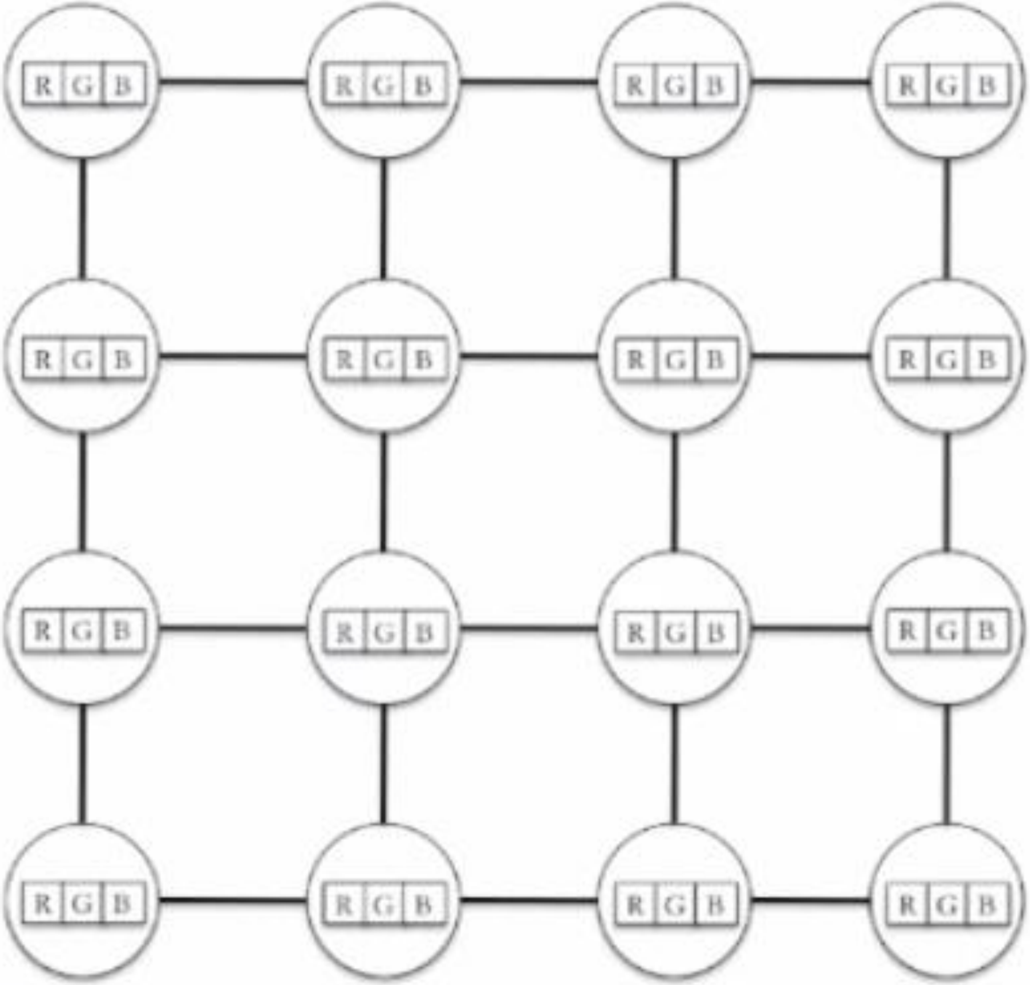
```
1.000000
0.961635
0.924742
0.889264
0.855148
0.822340
0.790791
0.760453
0.731278
0.703222
0.676243
0.650299
0.625351
0.601359
0.578288
0.556102
0.534767
0.514251
0.494522
0.475549
0.457305
0.439761
```

Colour Example



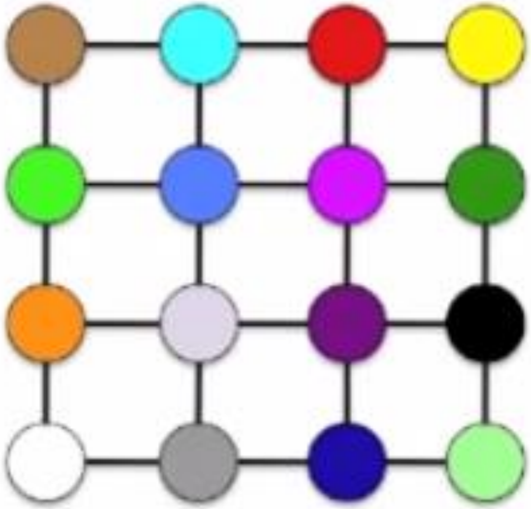
$$\begin{array}{ccc} & \begin{matrix} \text{R} & \text{G} & \text{B} \end{matrix} & \\ x = & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} & \begin{matrix} \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ \circ \end{matrix} \end{array}$$

Grid



Initialize Randomly

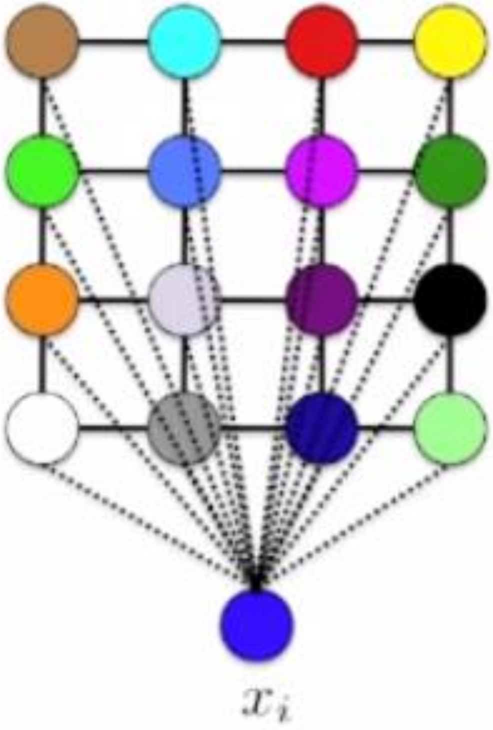
- Initialize each node with random features (colors)



Compare with Input

- Compare each input with all nodes in the grid

$$D_{ij} = \sqrt{\sum_{k=1}^{m} (x_{ik} - N_{jk})^2}$$



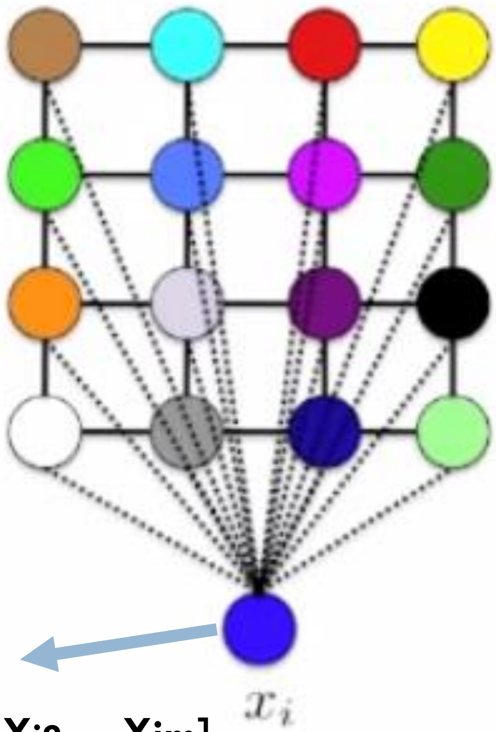
Compare with Input

- Compare each input with all nodes in the grid

$$D_{ij} = \sqrt{\sum_{k=1}^{m} (x_{ik} - N_{jk})^2}$$

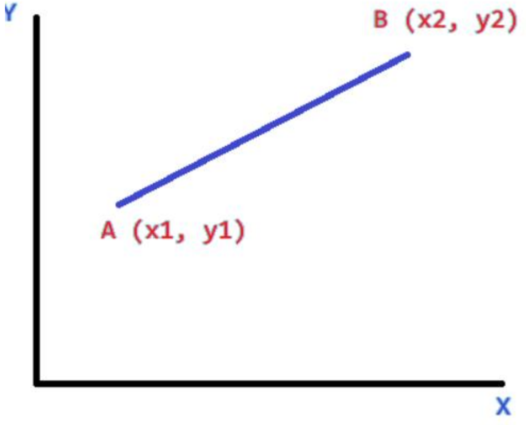
Vector

$N_j=[N_{j0},N_{j1},N_{j2}\dots.N_{jm}]$



Vector

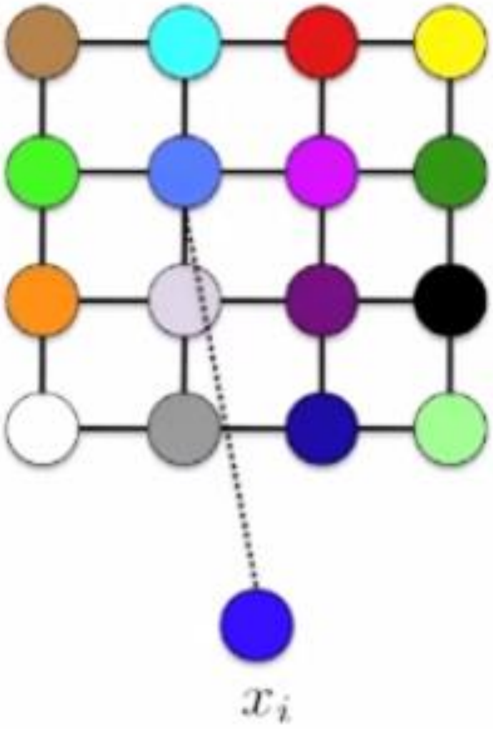
$X_i=[X_{i0},X_{i1},X_{i2}\dots.X_{im}]$



$$AB = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Best Matching Unit

- Select the node with minimum distance or the best matching unit (BMU)

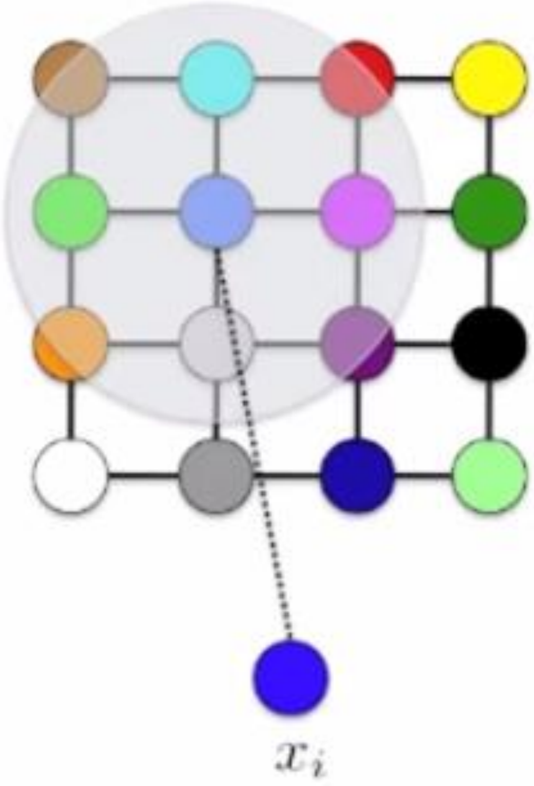


Update

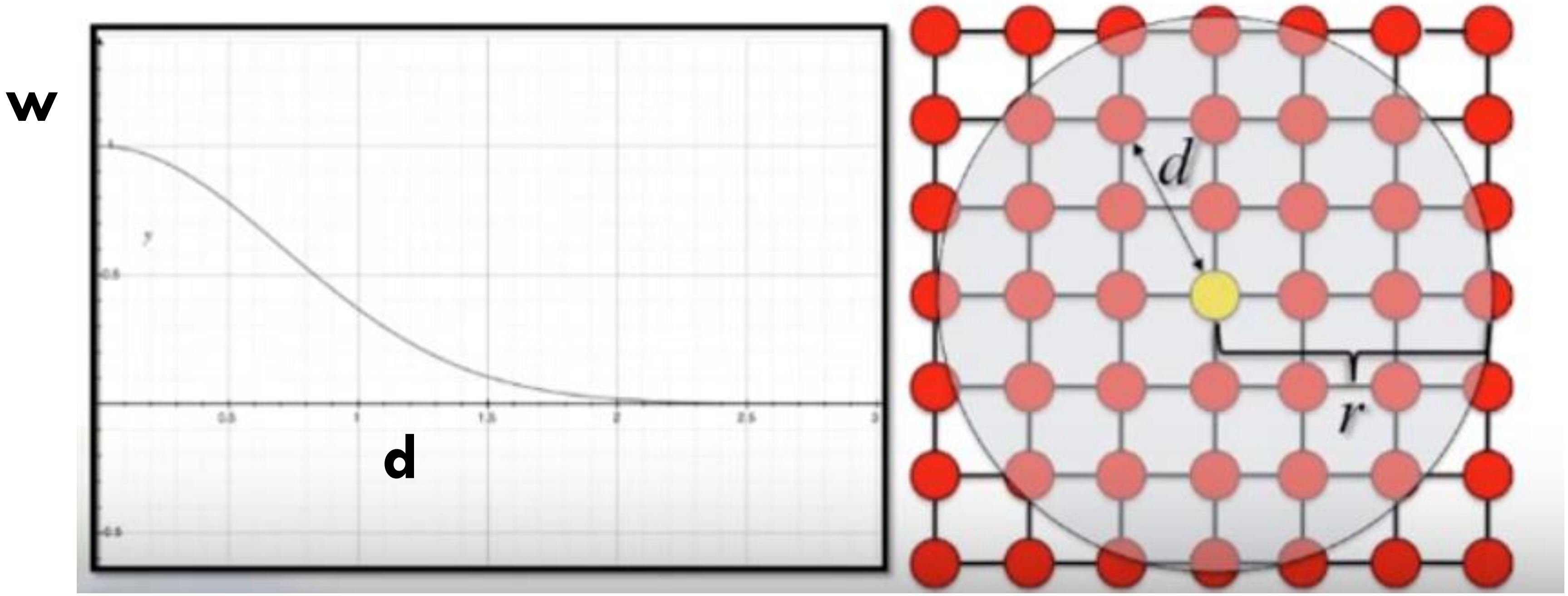
- For each node in the range of the BMU

$$N_j = N_j + \eta w_j \left(x_i - N_j\right)$$

- Where w_j is a weight parameter that depends on the distance between the node and the BMU and η is the learning rate

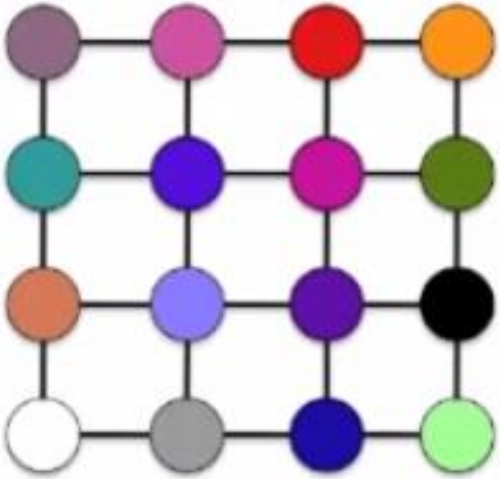


$$w_j = e^{-\frac{d_j^2}{2r^2}}$$



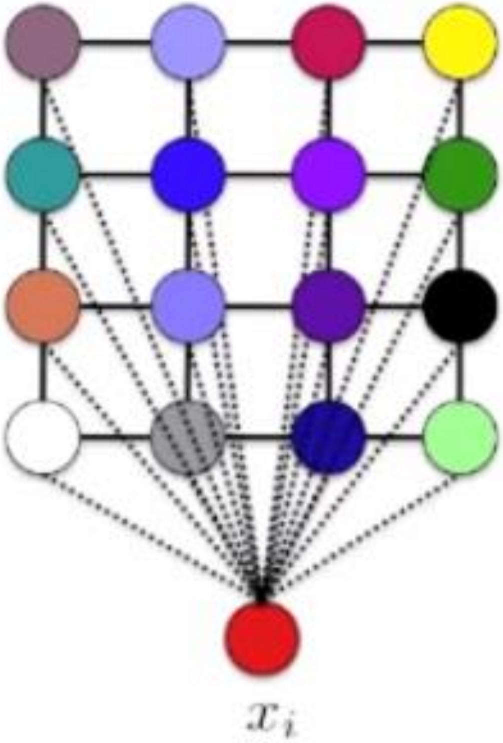
Compare with Input

- Repeat the process with different input



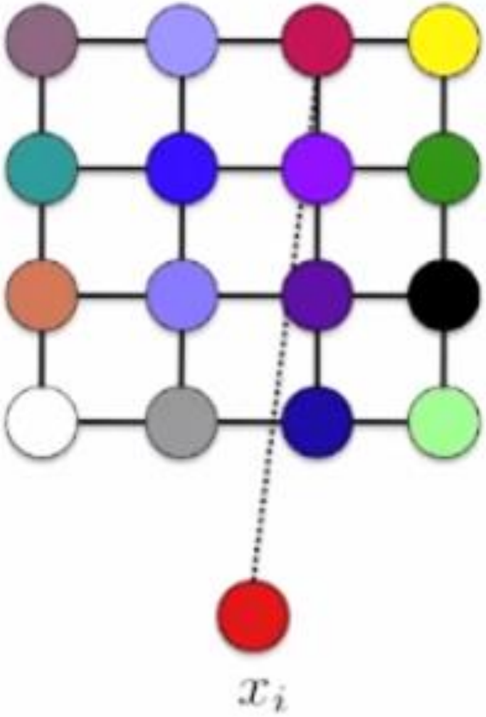
Compare with Input

- Repeat the process with different input

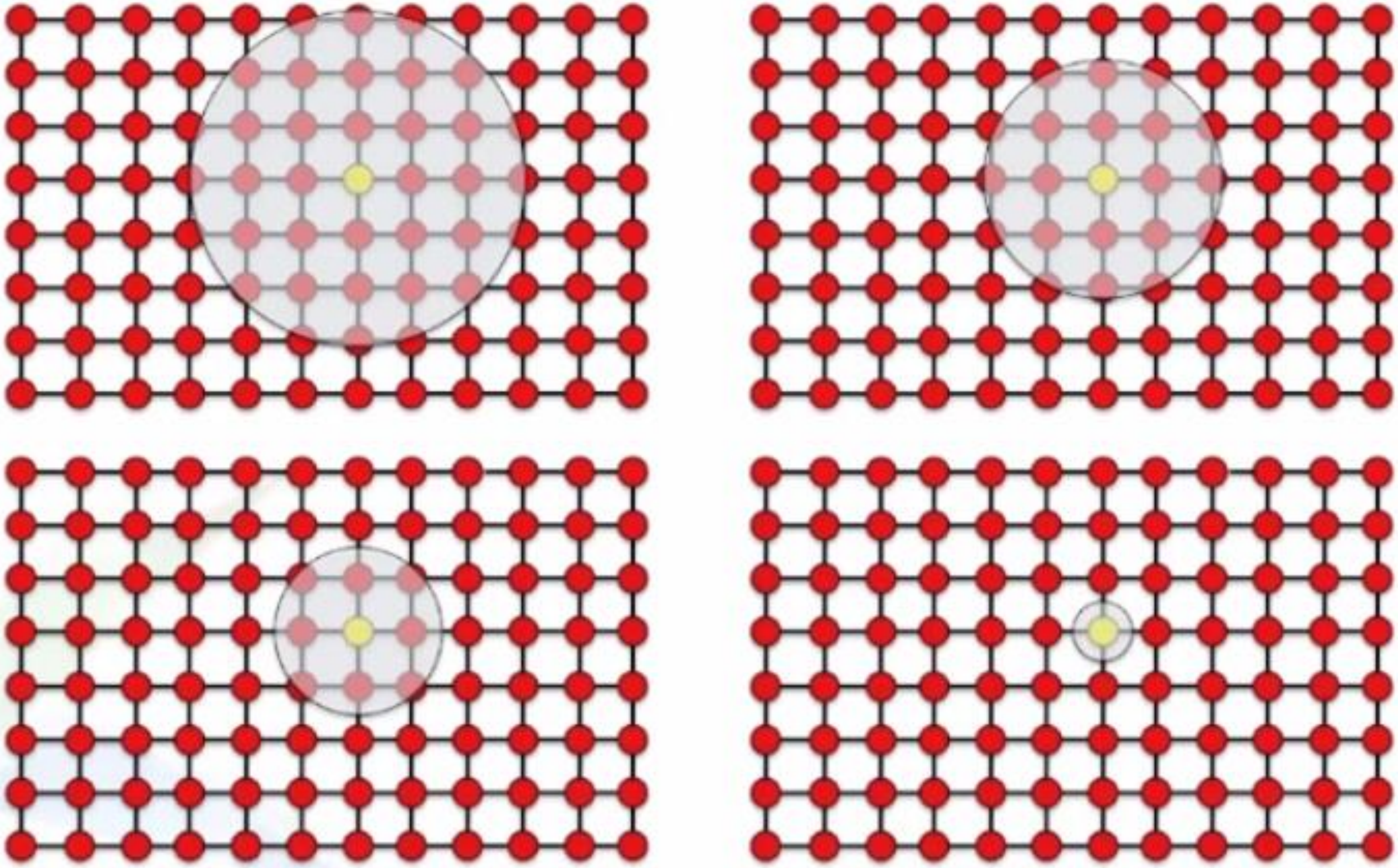


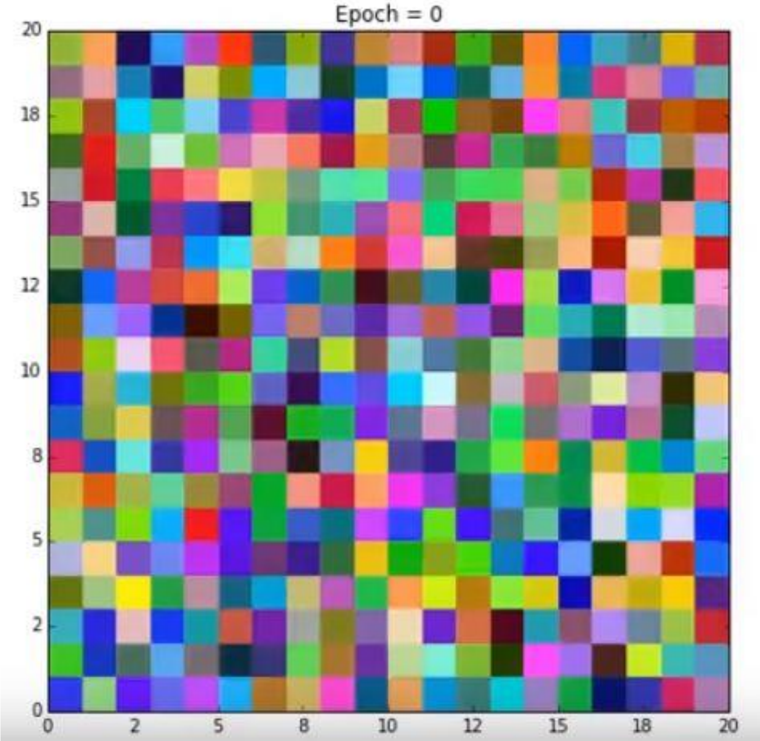
Compare with Input

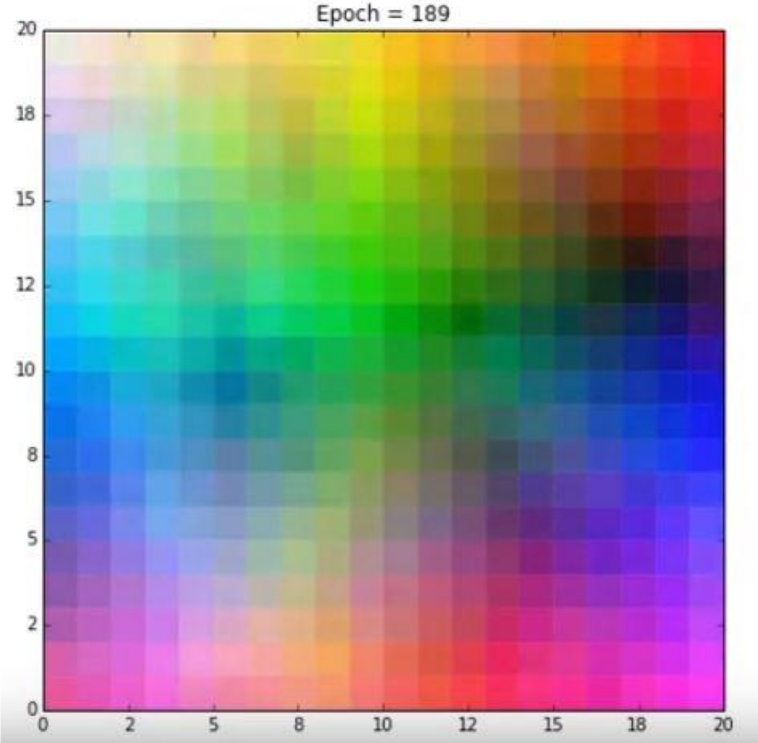
- Repeat the process with different input

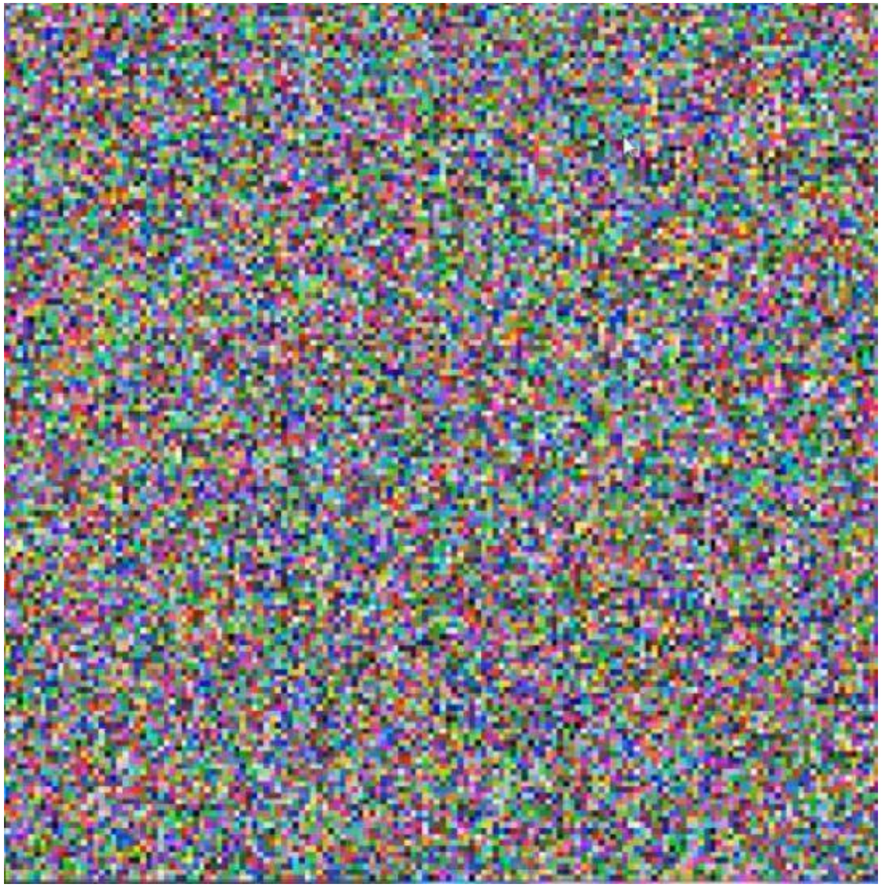


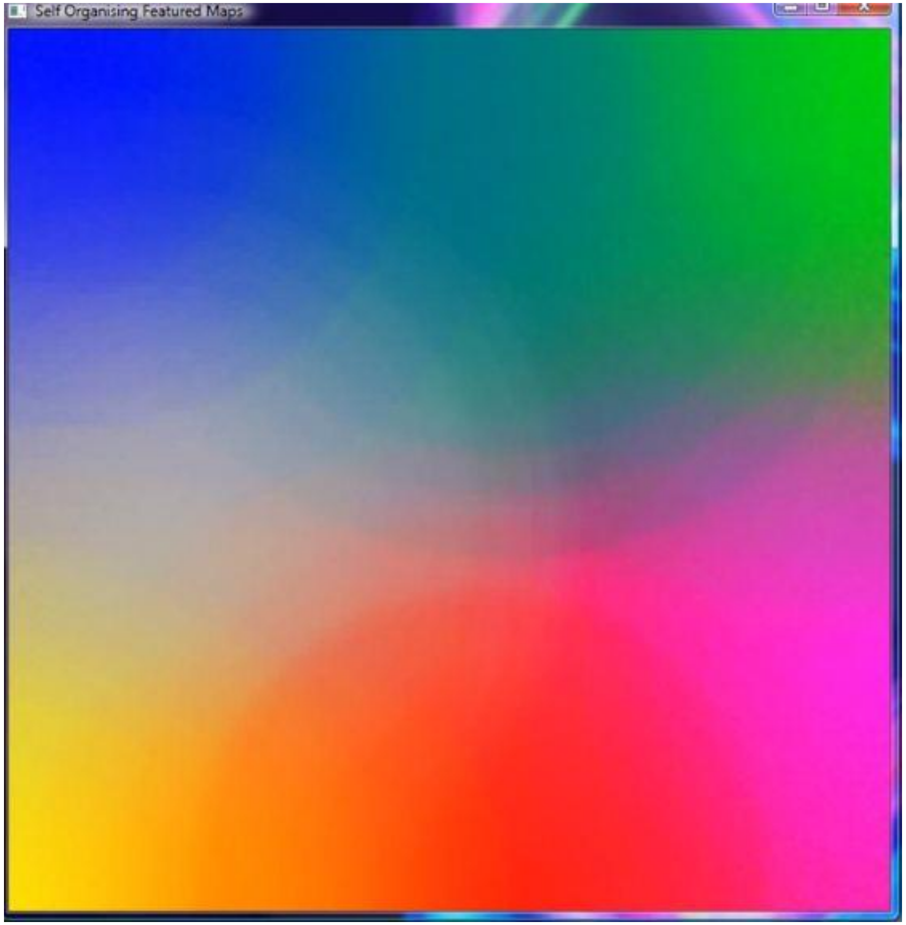
Repeat with smaller radius




Epoch = 0
20
18
15
12
10
8
5
2
0
0
2
5
8
10
12
15
18
20


Epoch = 189
20
18
15
12
10
8
5
2
0
0
2
5
8
10
12
15
18
20




Self Organising Featured Maps