Self Organizing Map

(자기 조직화 지도)

21.01.06

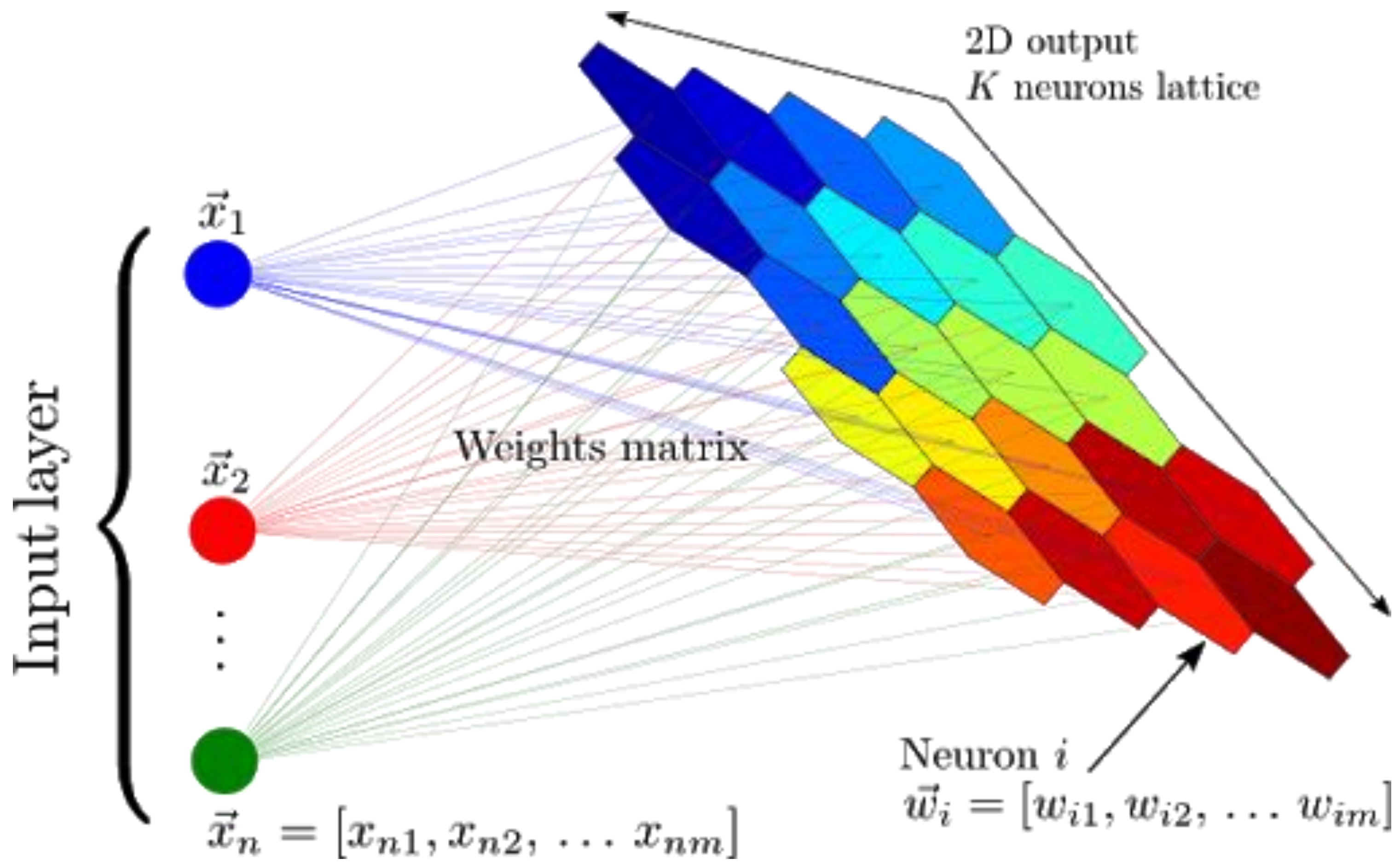
Seunghan Lee

Contents

1. What is Self Organizing Map (SOM) ?

- Unsupervised Learning
- Clustering on “Grid”
- Key :
  - (1) Dimension reduction
  - (2) Clustering

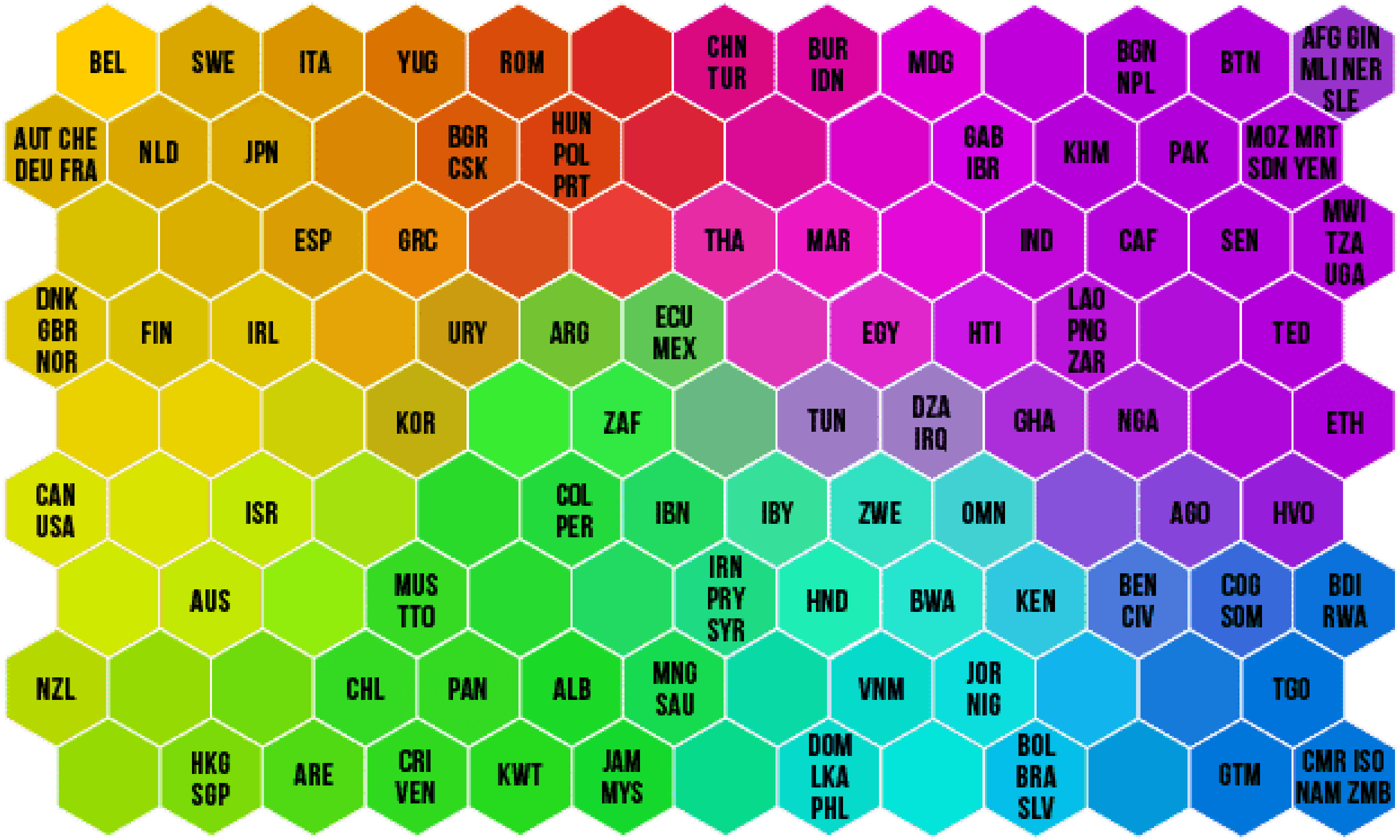
1. What is Self Organizing Map (SOM) ?



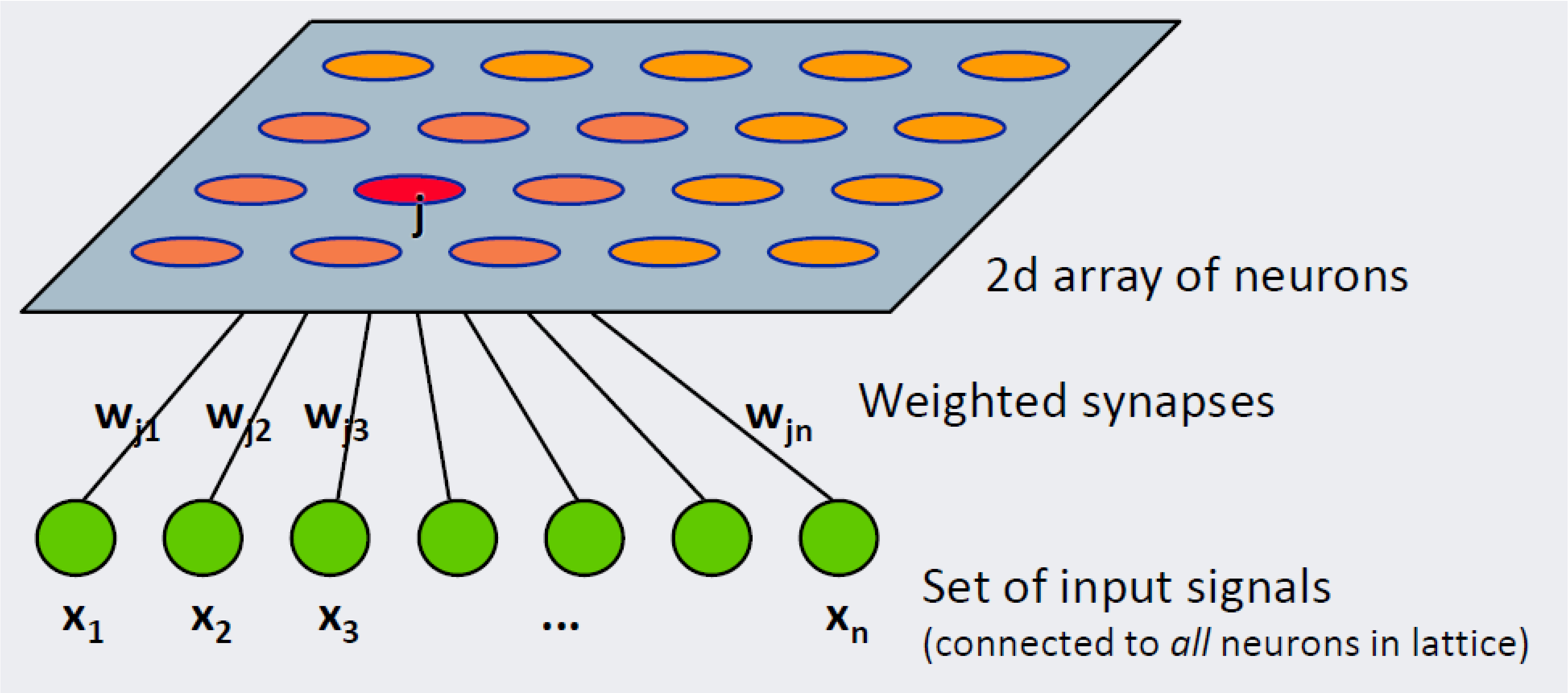
2. Architecture of SOM

- Visualizing high-dimensional data to low-dimensional space (usually 2D(or 3D) space)
- Use the algorithm based on Neural Net
- Similar cluster -> close to each other in the grid

2. Architecture of SOM

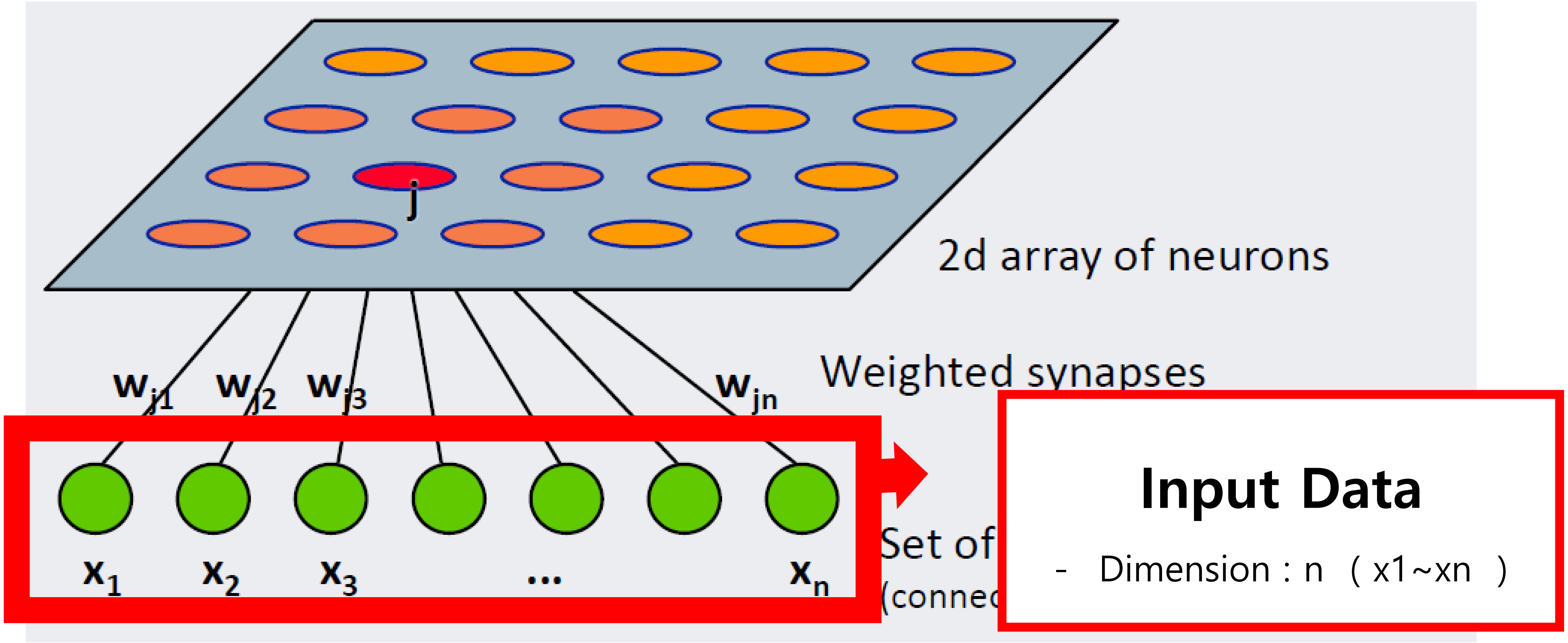


2. Architecture of SOM



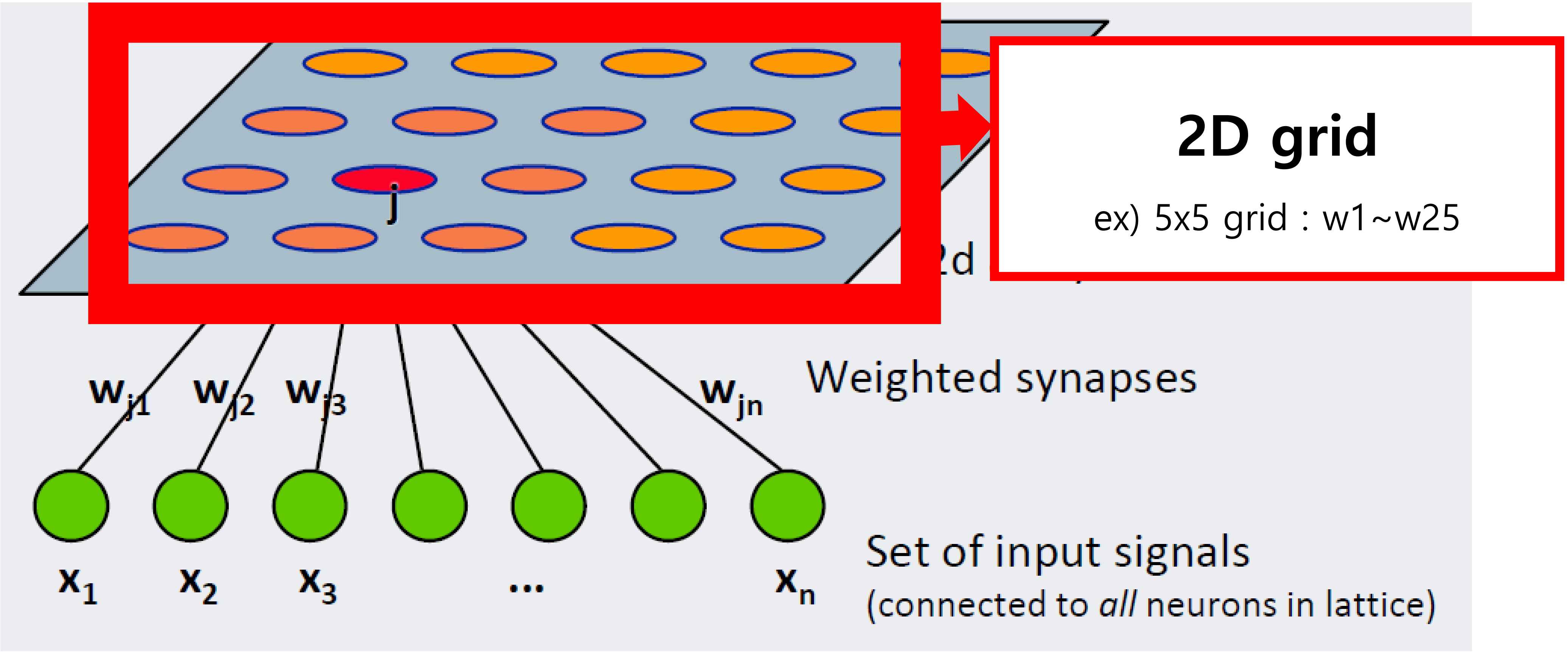
(source : https://ratsgo.github.io/machine%20learning/2017/05/01/SOM/)

2. Architecture of SOM



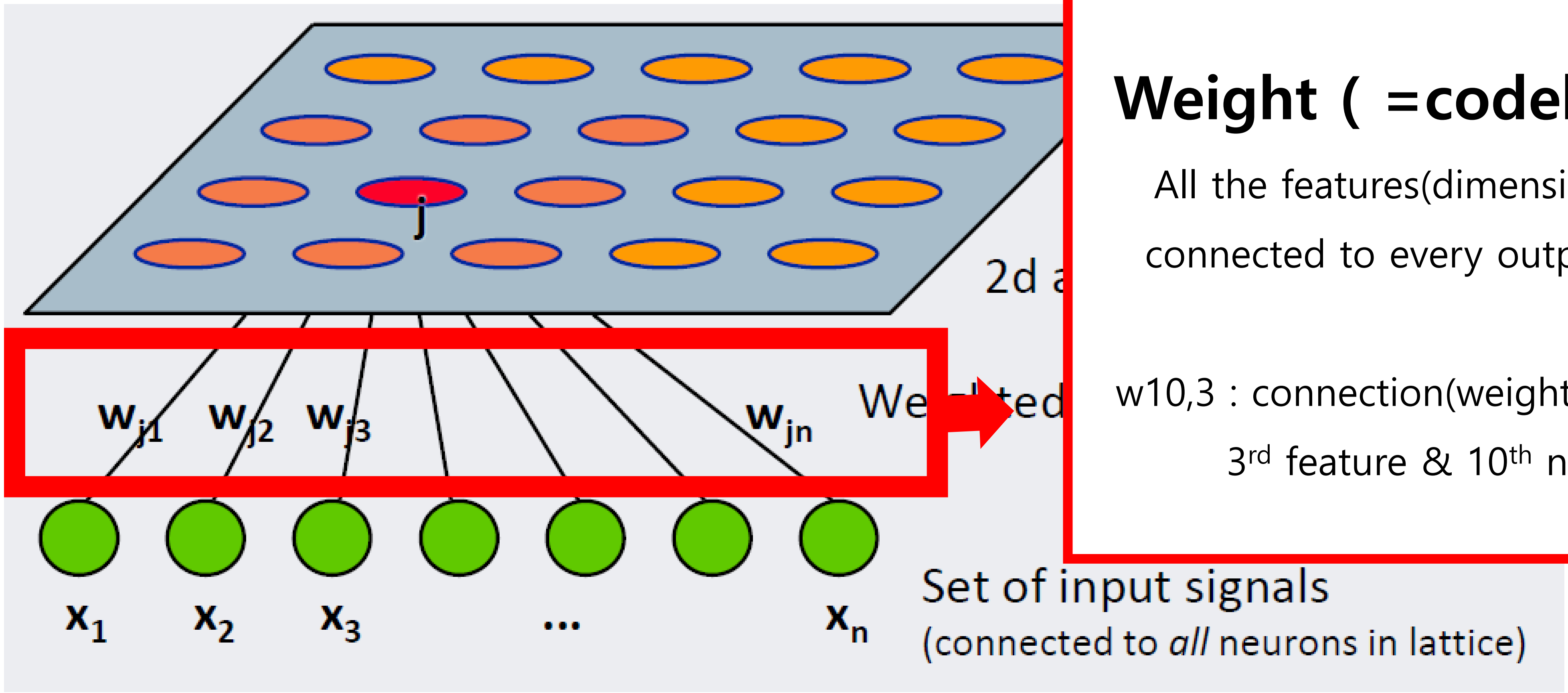
(source : https://ratsgo.github.io/machine%20learning/2017/05/01/SOM/)

2. Architecture of SOM



(source : https://ratsgo.github.io/machine%20learning/2017/05/01/SOM/)

2. Architecture of SOM



Weight (=codebook)

All the features(dimensions) are connected to every output node

w10,3 : connection(weight) between 3rd feature & 10th node

(source : https://ratsgo.github.io/machine%20learning/2017/05/01/SOM/)

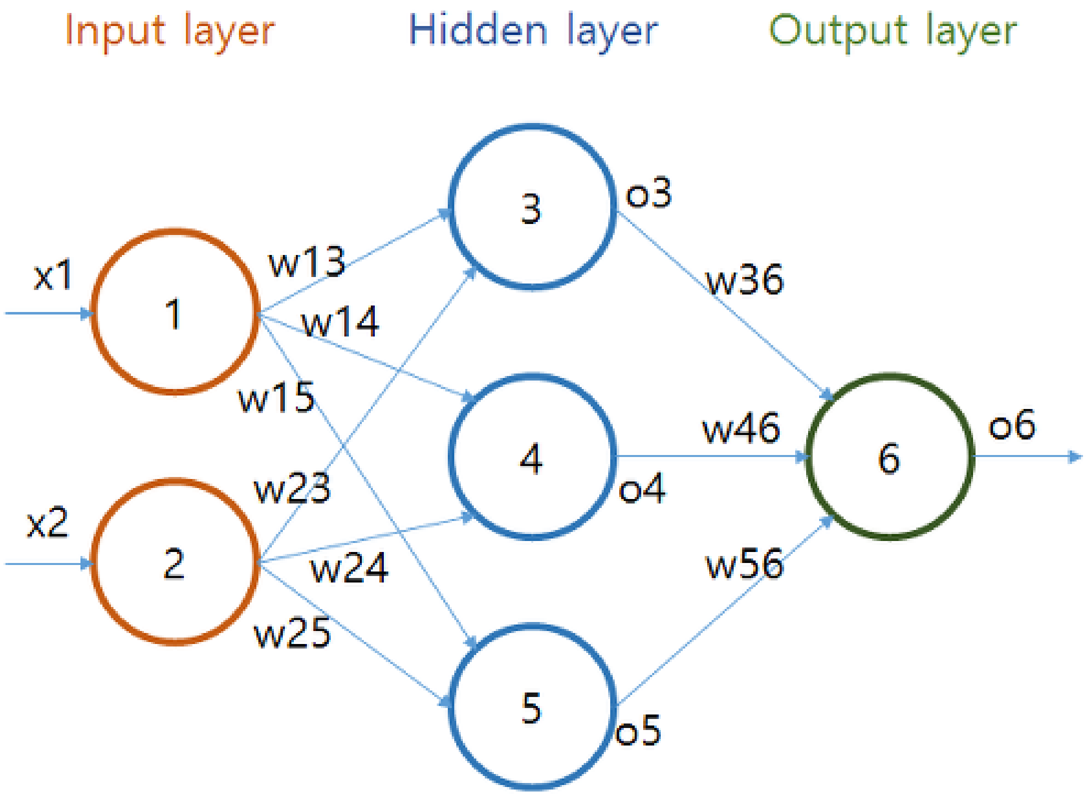
3. Basic of Neural Network

- How to Train SOM ? = (1) + (2)
  - (1) How to reduce dimension of our data?
  - (2) How to do clustering?

Have to know the about Neural Network!

(How the models based on Neural Network are trained)

3. Basic of Neural Network



Step 1) Initialize the weights

Step 2) Feed forward (----->)

Step 3) Calculate Error

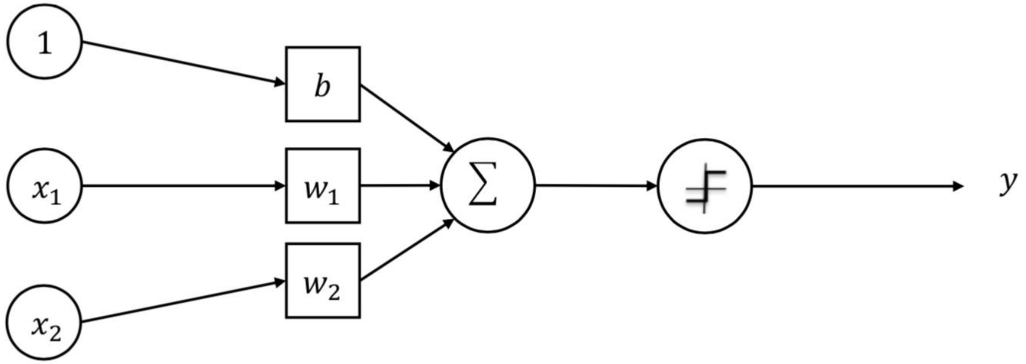
Step 4) Back Propagation (<-----)

Backpropgataion :

Updating the weights in a way that reduces the error (cost function)

3. Basic of Neural Network

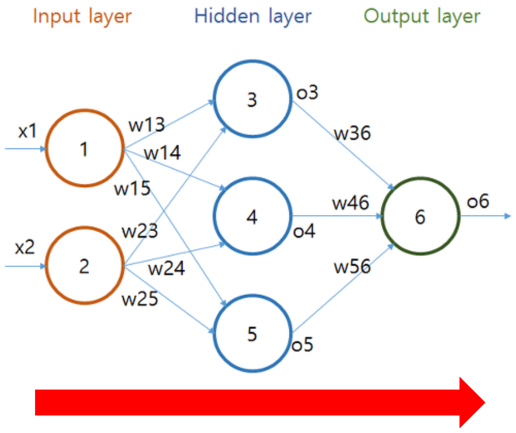
뉴런 Neuron



입력 (Input node) | 가중치와 편향 (Weights and bias) | 활성 함수 (Activation function) | 출력 (Output node)

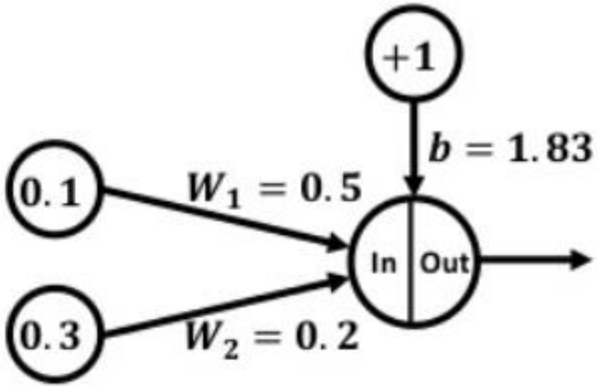
신경망은 뉴런을 기본 단위로 하며, 이를 조합하여 복잡한 구조를 이룬다.

3. Basic of Neural Network



Network Training

- Steps to train our network:
  1. Prepare activation function input (sum of products between inputs and weights).
  2. Activation function output.



1

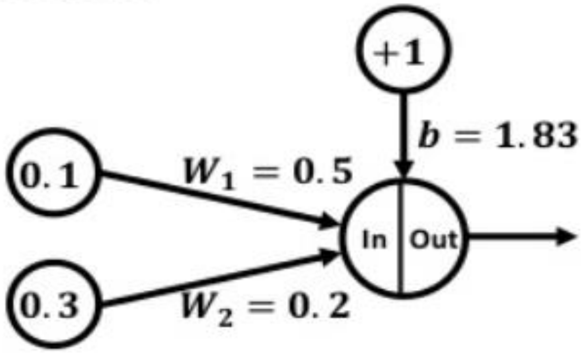
Network Training: Sum of Products

$$s = X_1 * W_1 + X_2 * W_2 + b$$

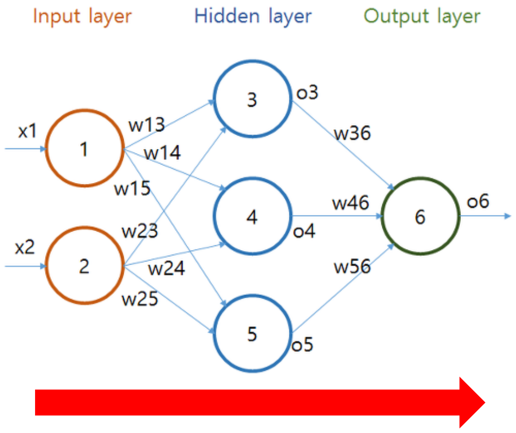
$$s = 0.1 * 0.5 + 0.3 * 0.2 + 1.83$$

$$s = 1.94$$

- After calculating the sop between inputs and weights, next is to use this sop as the input to the activation function.



(Source : https://www.slideshare.net/AhmedGadFCIT/backpropagation-understanding-how-to-update-anns-weights-stepbystep)



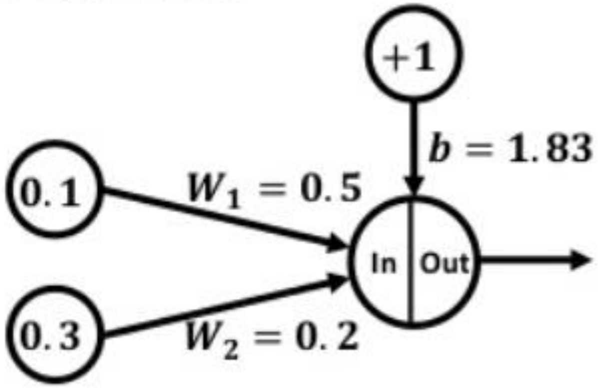
3. Basic of Neural Network

2

Network Training: Activation Function

- In this example, the sigmoid activation function is used.

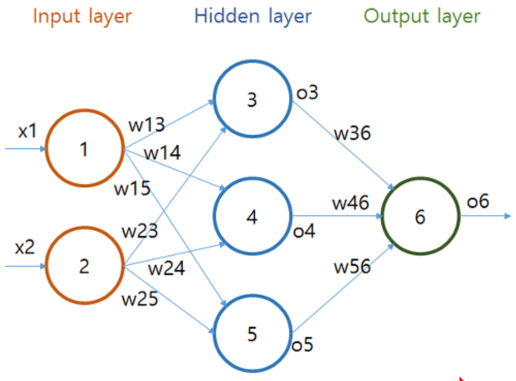
$$f(s) = \frac{1}{1+e^{-s}}$$



- Based on the sop calculated previously, the output is as follows:

$$f(s) = \frac{1}{1+e^{-1.94}} = \frac{1}{1+0.144} = \frac{1}{1.144}$$

$$f(s) = 0.874$$



3. Basic of Neural Network

3

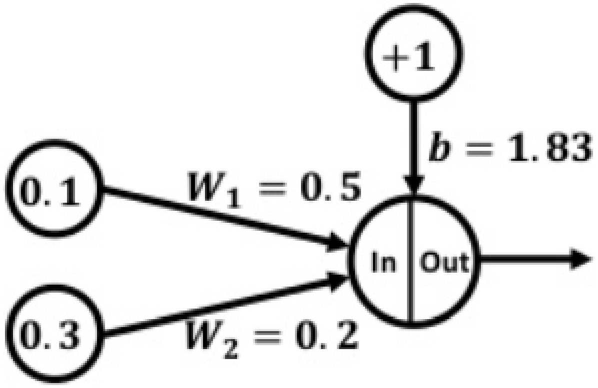
Network Training: Prediction Error

- After getting the predicted outputs, next is to measure the **prediction error** of the network.
- We can use the squared error function defined as follows:

$$E = \frac{1}{2}(desired - predicted)^2$$

- Based on the predicted output, the prediction error is:

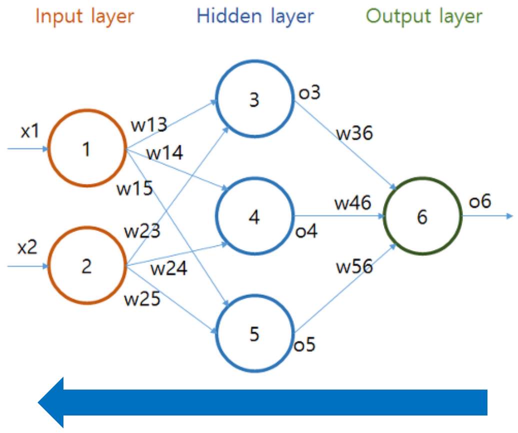
$$E = \frac{1}{2}(0.03 - 0.874)^2 = \frac{1}{2}(-0.844)^2 = \frac{1}{2}(0.713) = 0.357$$



Finished Making a prediction! But... too large error!

Have to update(change) our weight, that can make error smaller!

3. Basic of Neural Network



How to update our weights?

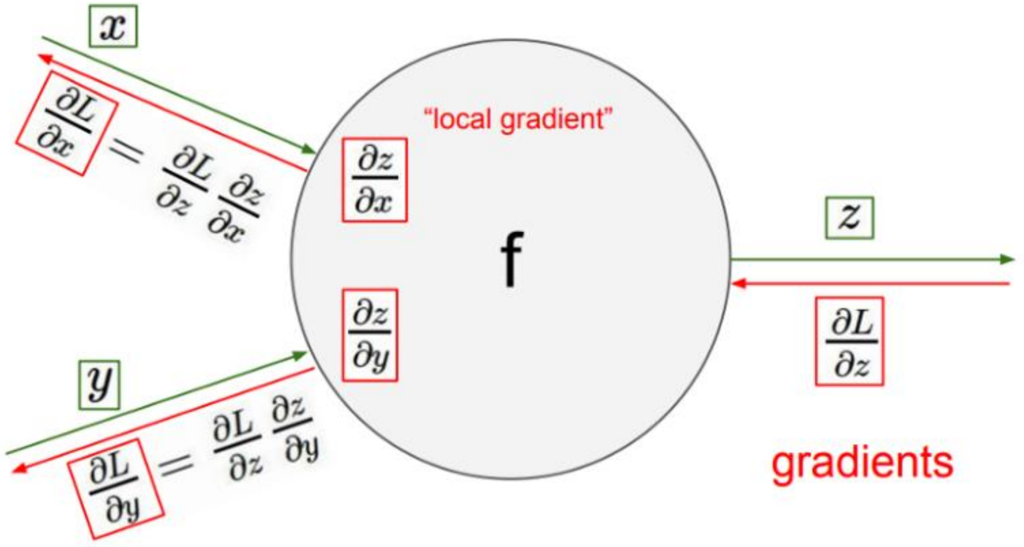
Derivative of Error with respect to weight

Old weight

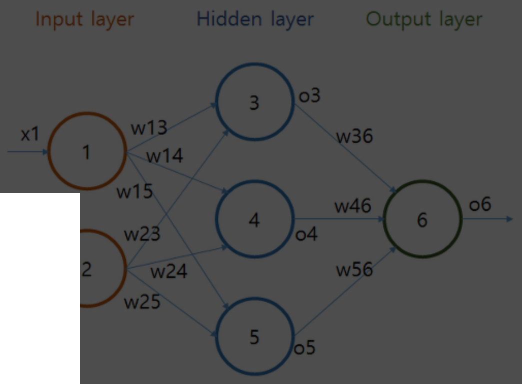
$$^{*}W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right)$$

New weight

Learning rate



$$\boxed{\frac{\delta(Etotal)}{\delta w5} = \frac{\delta(Etotal)}{\delta Output_{OL1}} * \frac{\delta(Output_{OL1})}{\delta Input_{OL1}} * \frac{\delta Input_{OL1}}{\delta w5}}$$



2. Ba

How to u

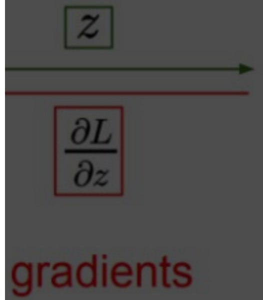
Derivative of Error with respect to weight

Old weight

$$^{*}W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right)$$

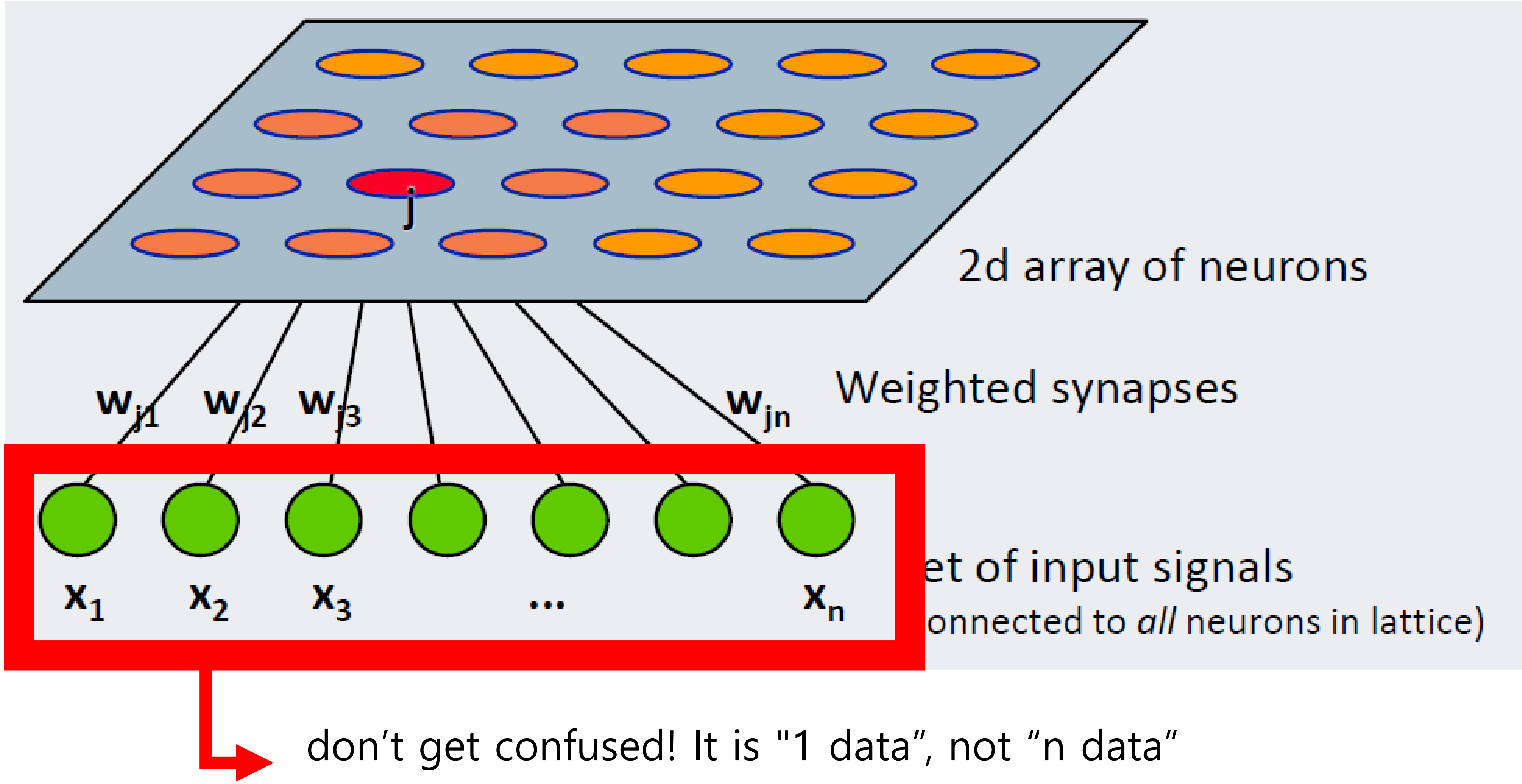
New weight

Learning rate



$$\boxed{\frac{\delta(Etotal)}{\delta w5} = \frac{\delta(Etotal)}{\delta Output_{OL1}} * \frac{\delta(Output_{OL1})}{\delta Input_{OL1}} * \frac{\delta Input_{OL1}}{\delta w5}}$$

4. Training SOM

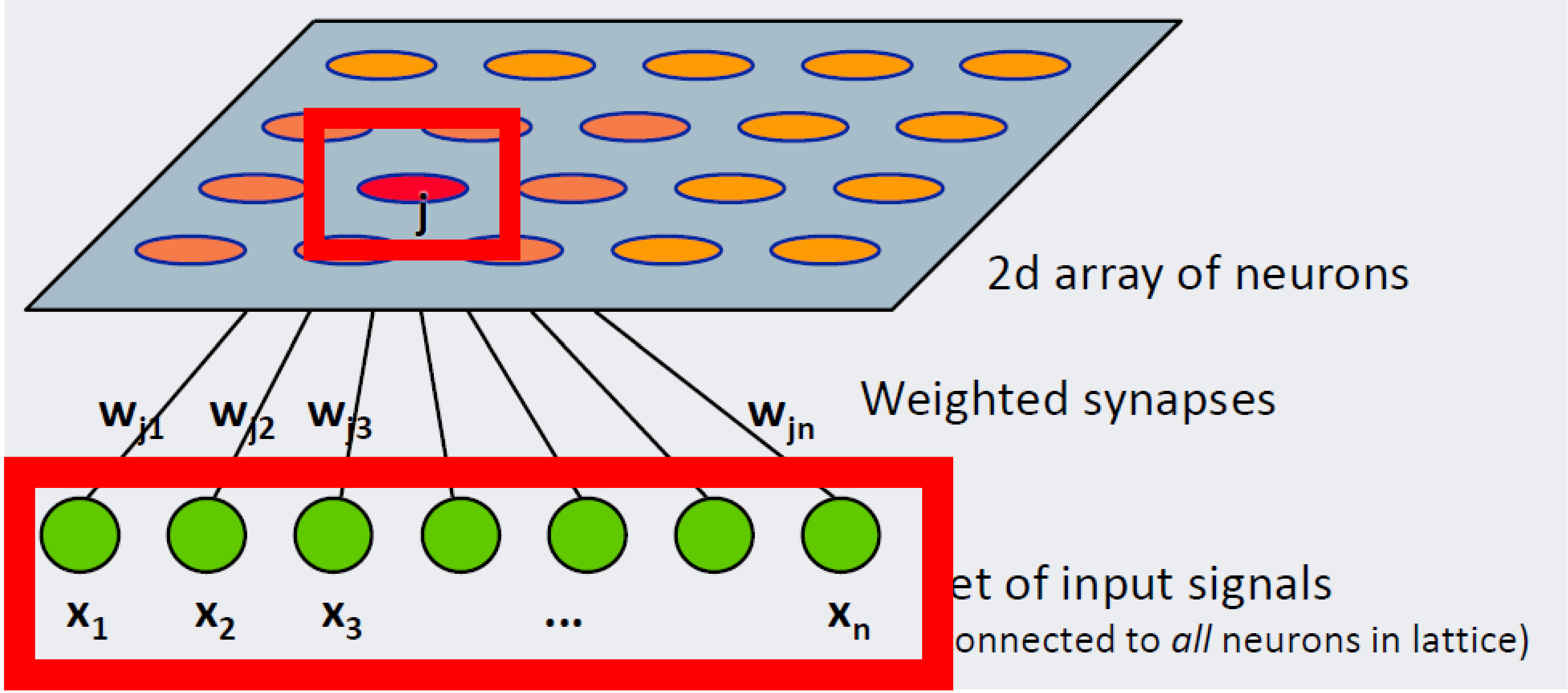


don’t get confused! It is "1 data”, not “n data”

("1 data“ with "n dimension“)

4. Training SOM

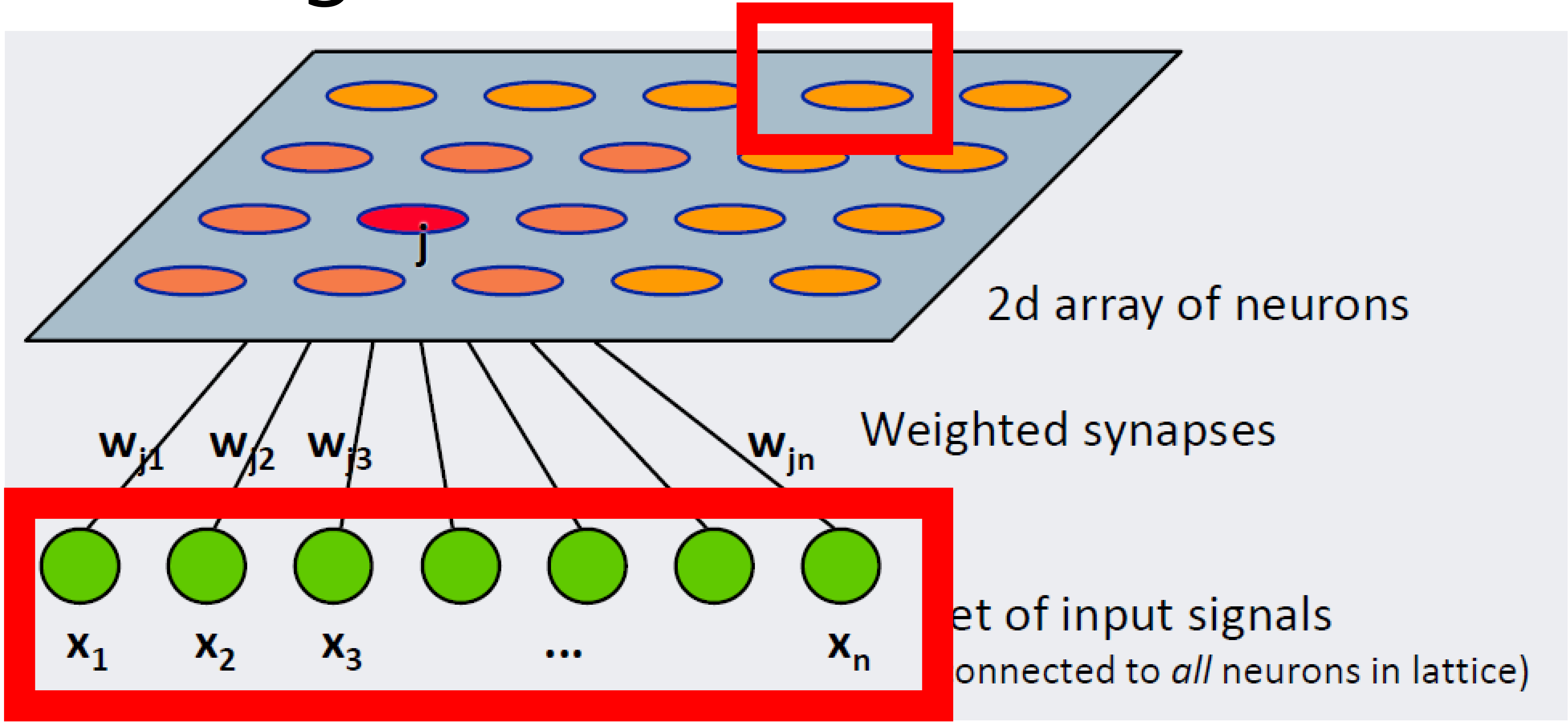
Data 1



Each data is assigned to one of the nodes!

4. Training SOM

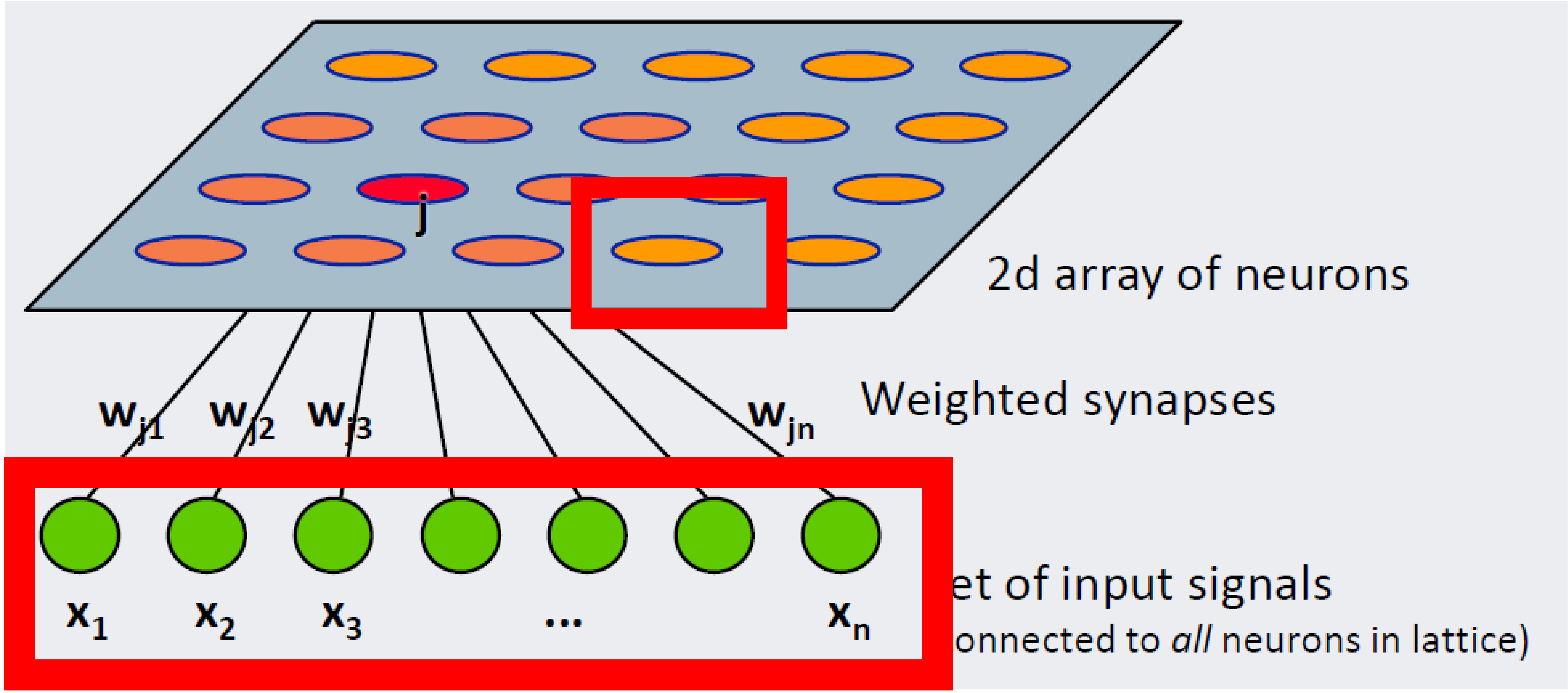
Data 2



Each data is assigned to one of the nodes!

4. Training SOM

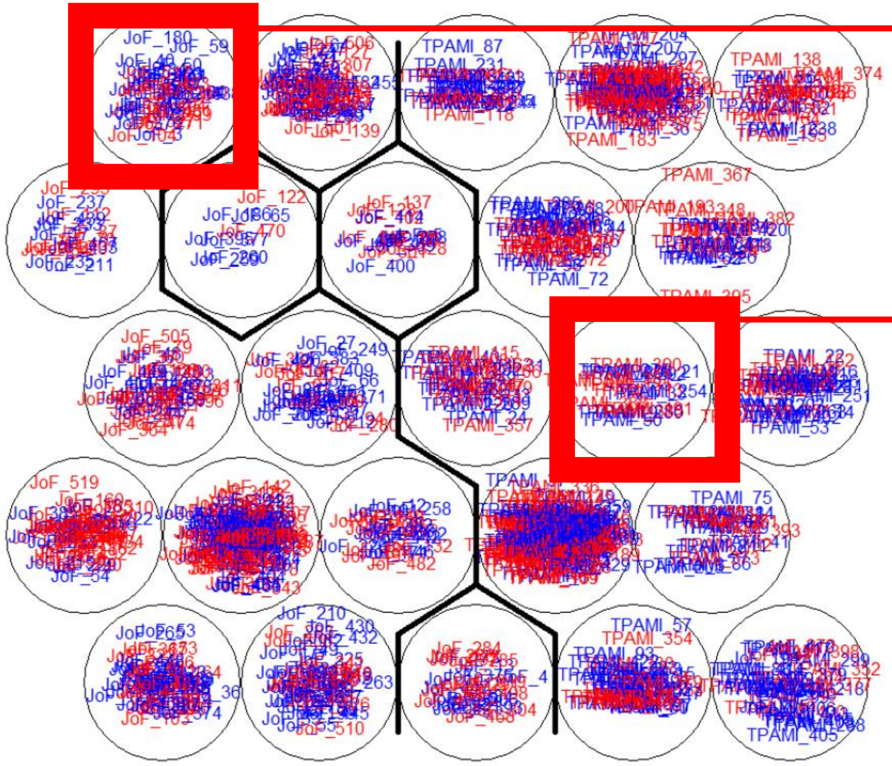
Data 3



Each data is assigned to one of the nodes!

4. Training SOM

After all assignments are made….



Cluster 1
(= Cluster (1,1))

Cluster 14
(= Cluster (3,4))

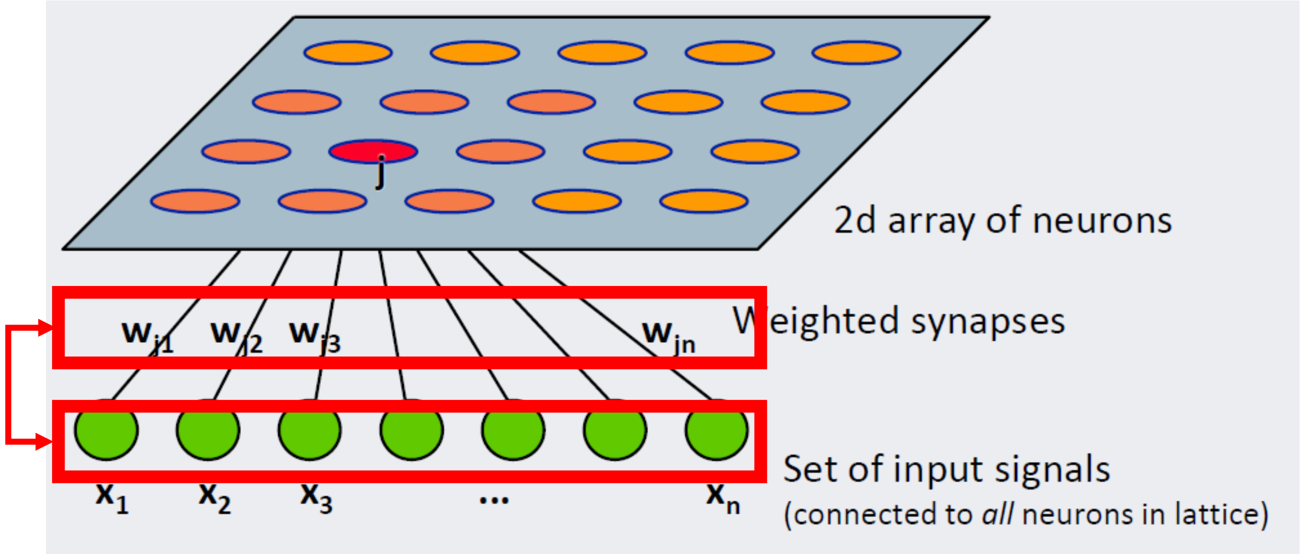
4. Training SOM

To which neurons will each data be assigned?

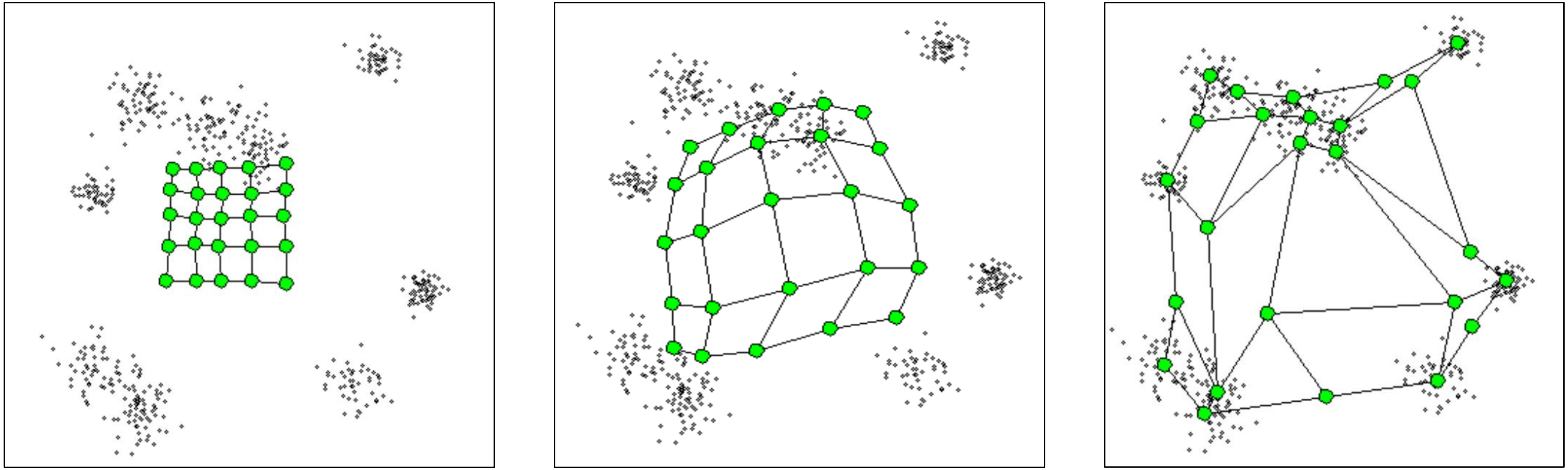
- Assign to the closest node (ex. among 25 nodes!)

(closest node = "winning node" / "BMU(Best Matching Unit)

- ex) Euclidean Distance

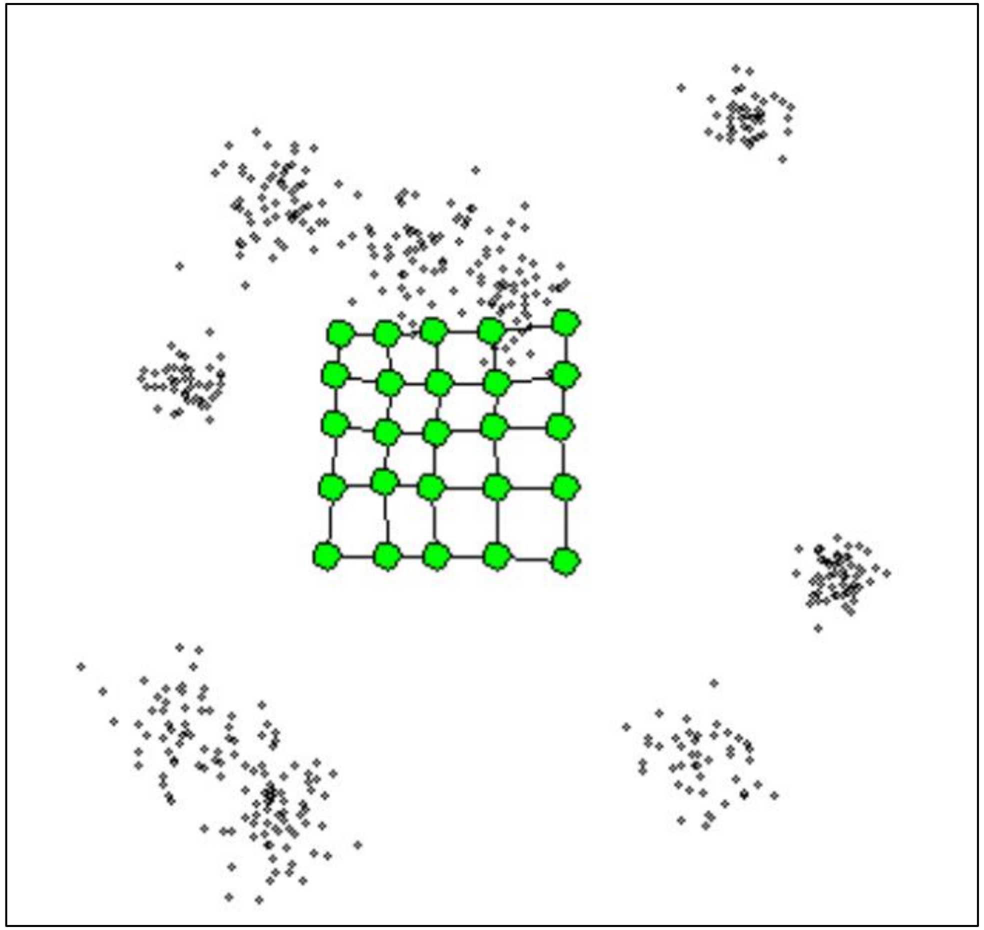


4. Training SOM



How?

4. Training SOM



Black : our data

Green : grid nodes (neurons)

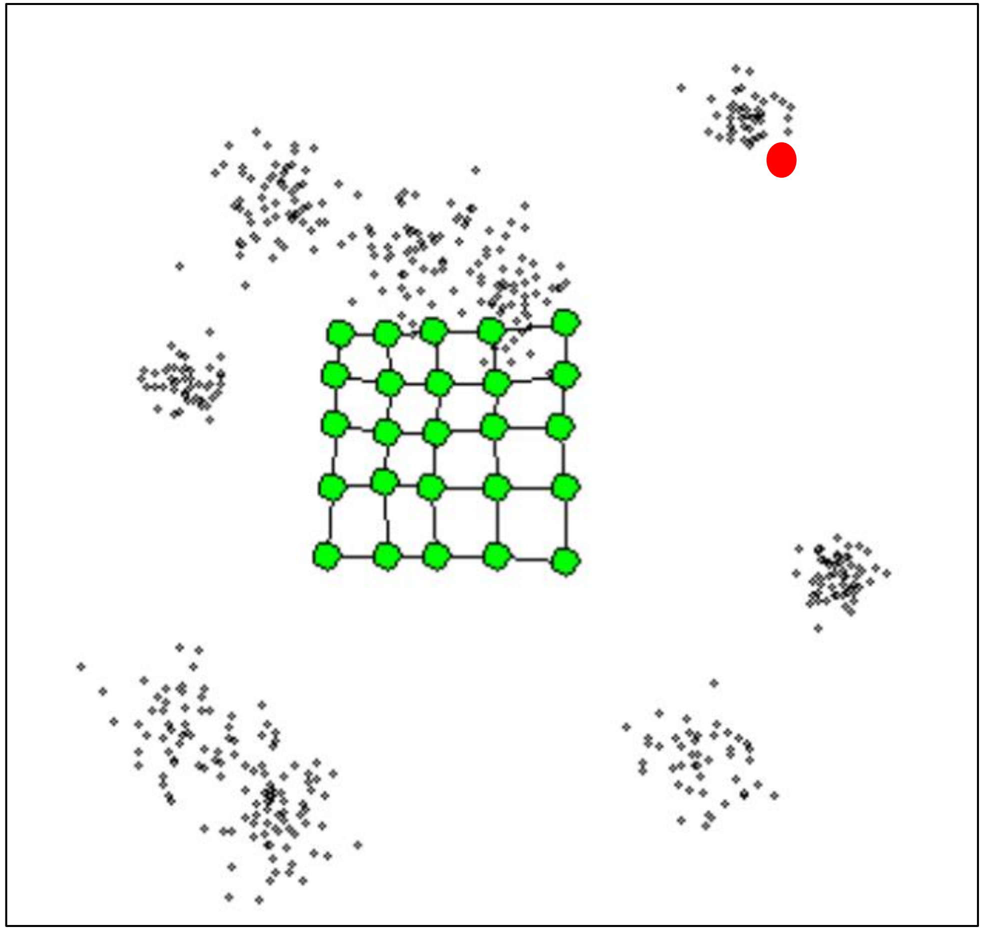
[Step1] Initialize **grid nodes** randomly

(= randomize weights (wij))

(Input x weights = grid node)

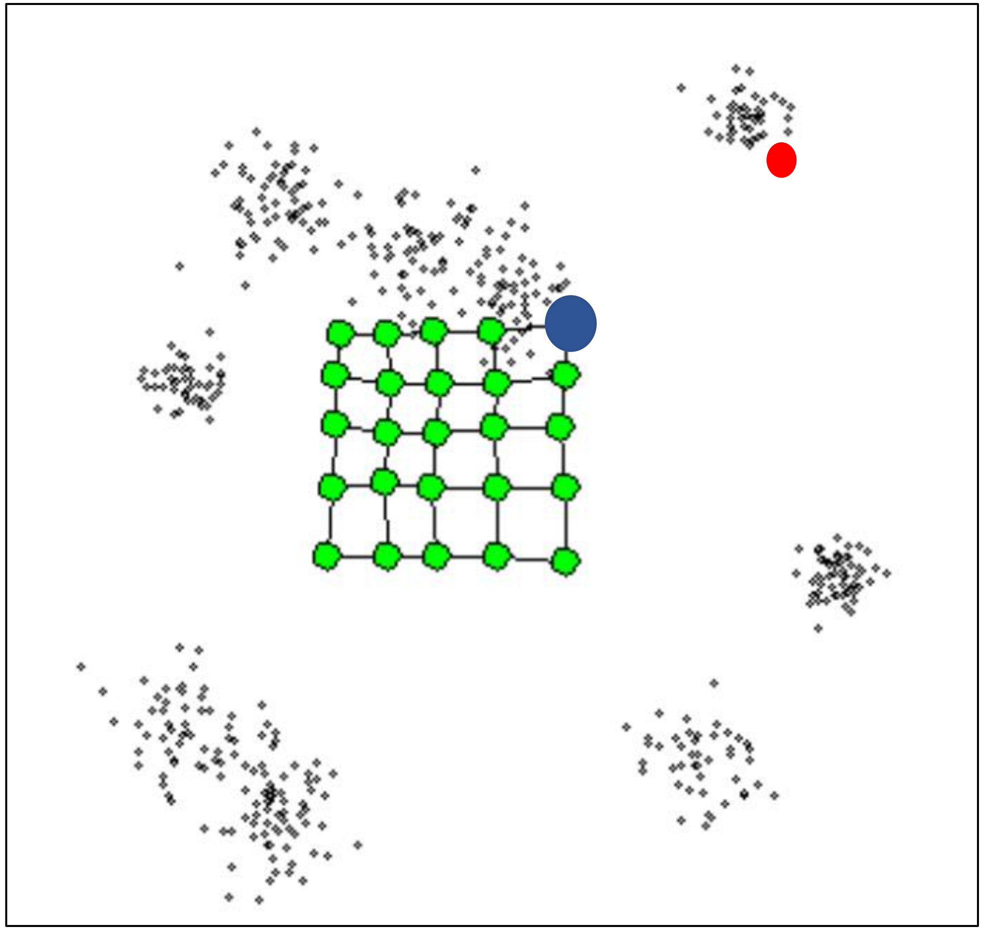
(source : https://ratsgo.github.io/machine%20learning/2017/05/01/SOM/)

4. Training SOM



[Step 2] Select one data randomly

4. Training SOM

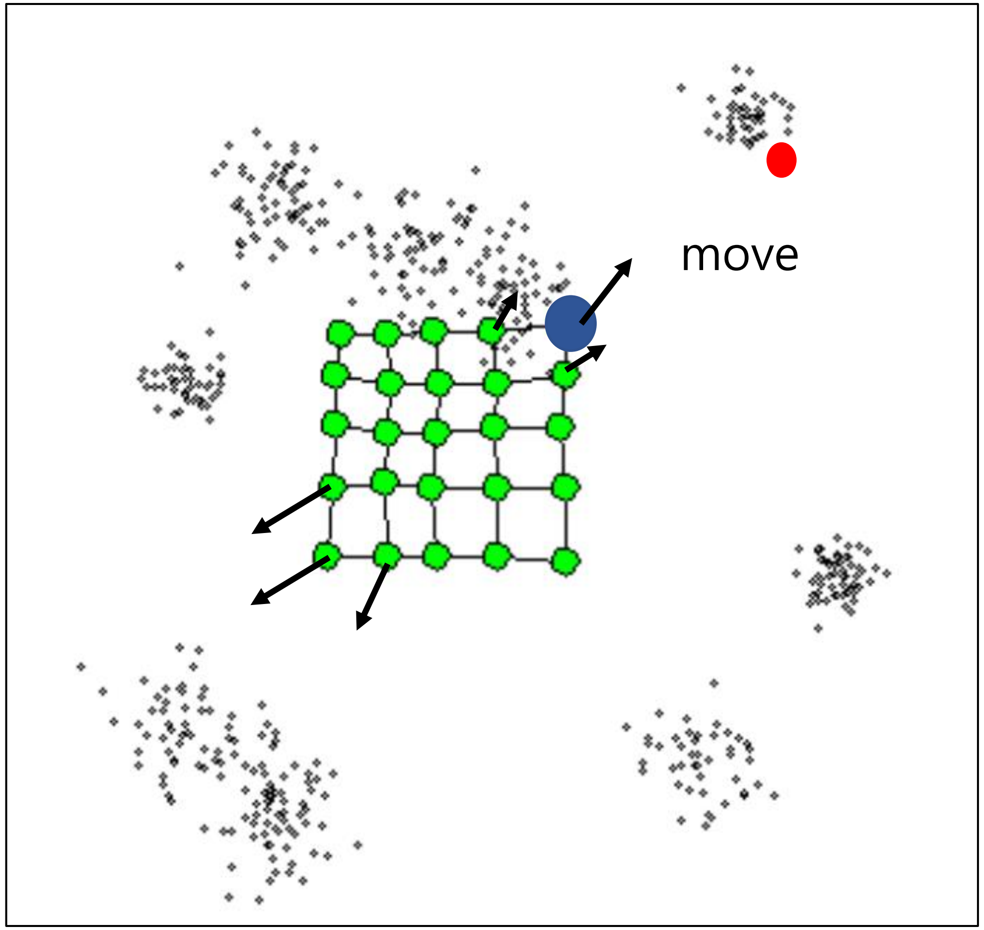


[Step 2] Select one data randomly

[Step 3] Find the **closest node** among grid nodes!

(closest node = **"winning node"**)

4. Training SOM



[Step 2] Select one data randomly

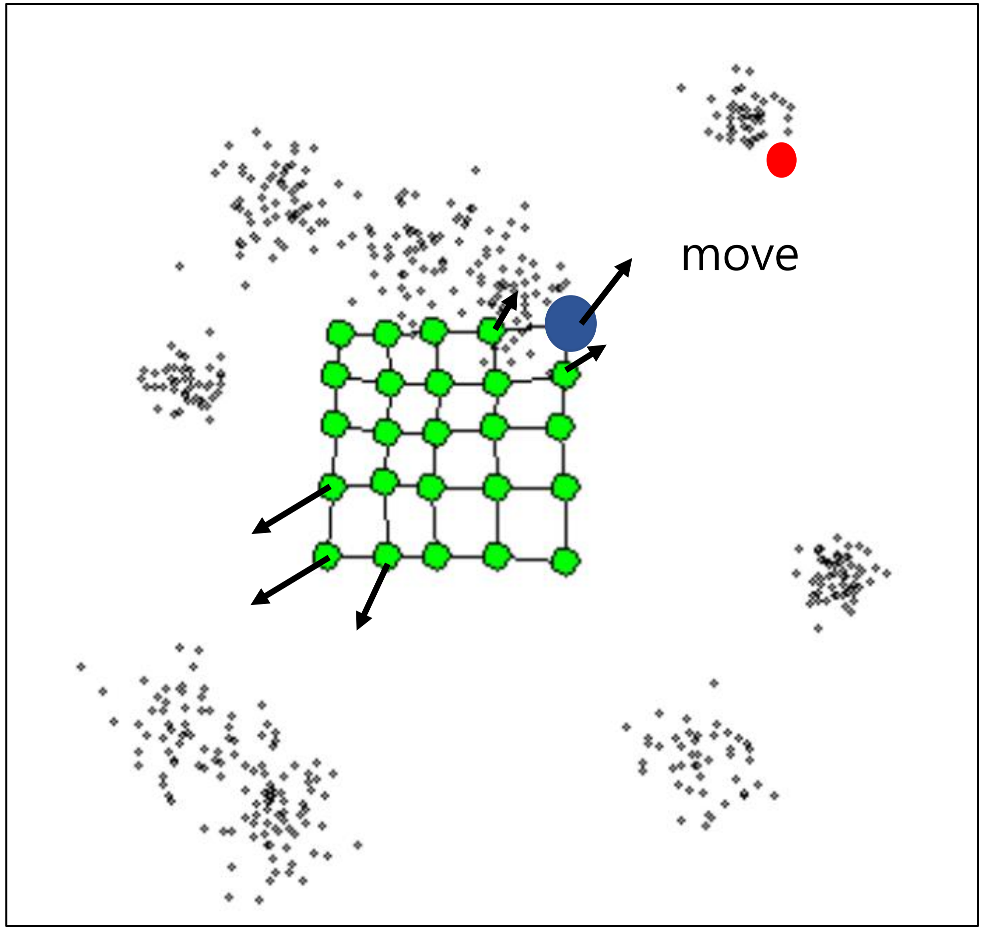
[Step 3] Find the **closest node** among grid nodes!

(closest node = **"winning node“**)

<u>[Step 4] Update..</u>

- 1) closet node (1)
- 2) other nodes (24)

4. Training SOM



[Step 2] Select one data randomly

[Step 3] Find the **closest node** among grid nodes!

(closest node = **"winning node"**)

[Step 4] Update..

- 1) closet node (1)
- 2) other nodes (24)

$$\boxed{w_j^{t+1} = w_j^t + \mu_t \lambda_x^{j,t} [x - w_j^t]}$$

4. Training SOM

$$\boxed{w_j^{t+1} = w_j^t + \boxed{\mu_t \lambda_x^{j,t} [x - w_j^t]}}$$

Weight at iteration "t+1"
(= new weight)

Weight at iteration "t"
(= old weight)

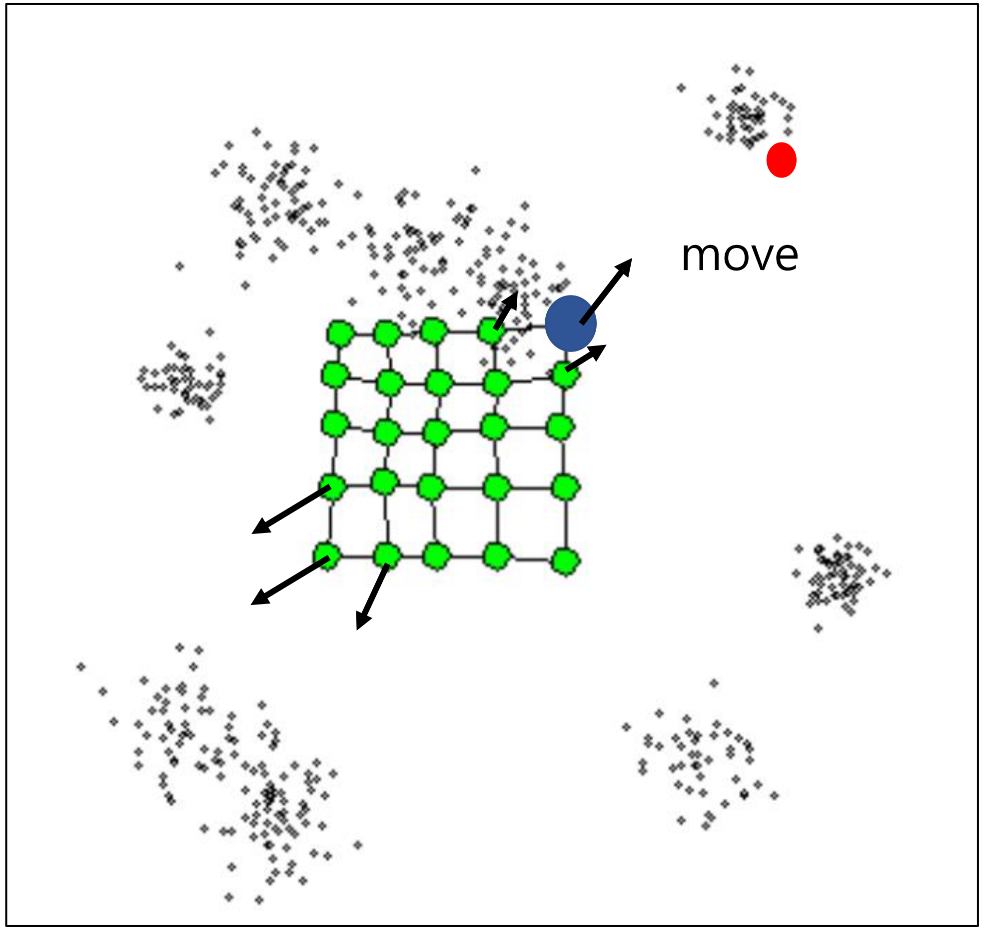
How much each weight changes

μ_t : Learning rate

$\lambda_x^{j,t}$: close node = big → change 'large'
far node = small → change 'little'

$[x - w_j^t]$: Difference between weight & input

4. Training SOM



[Step 2] Select one data randomly

[Step 3] Find the **closest node** among grid nodes!

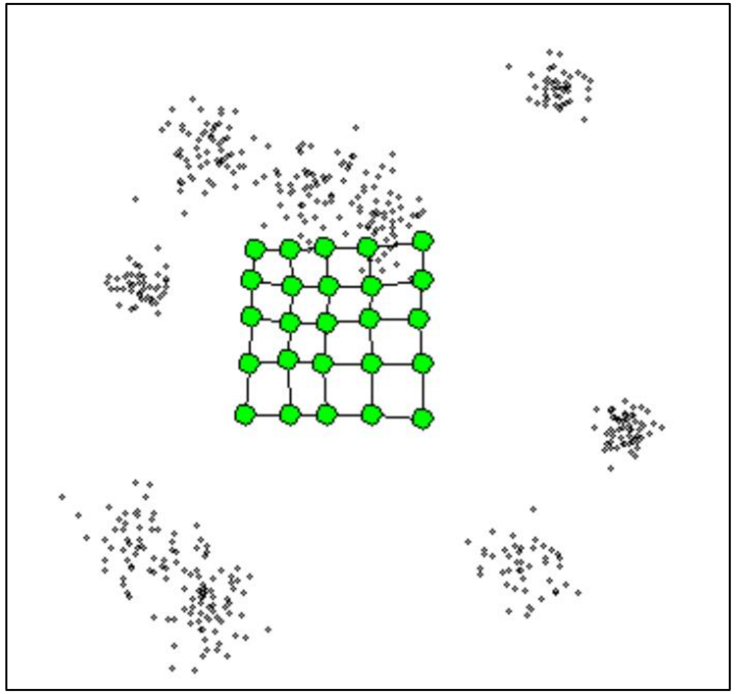
(closest node = **"winning node“**)

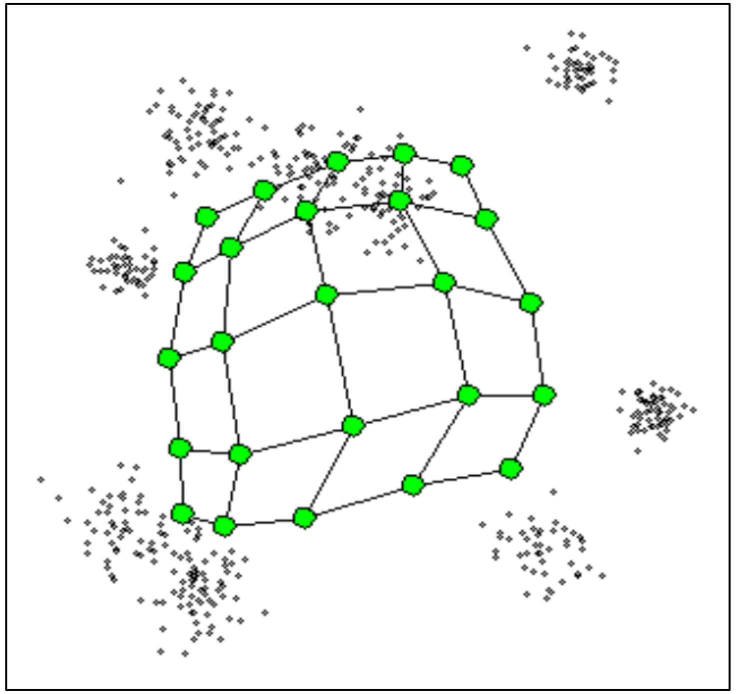
[Step 4] Update..

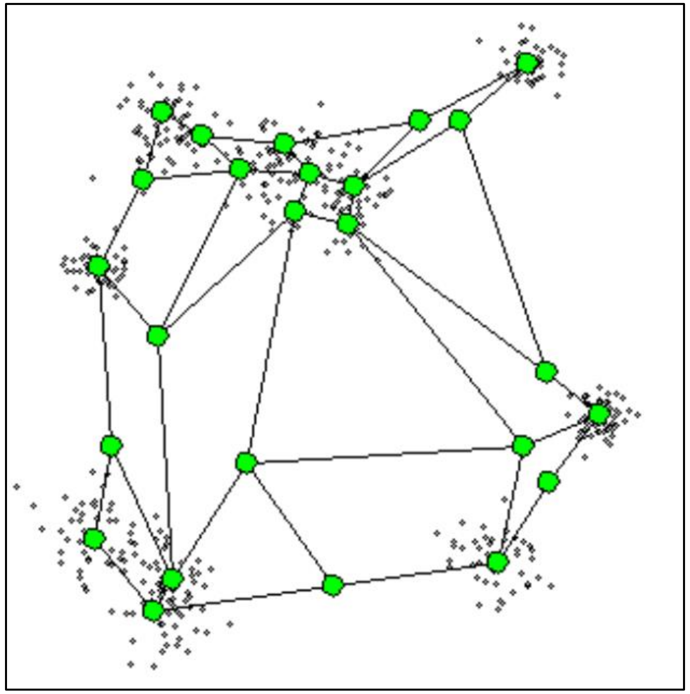
- 1) closet node (1)
- 2) other nodes (24)

<u>[Step 5] Gradually reduce the learning rate</u>

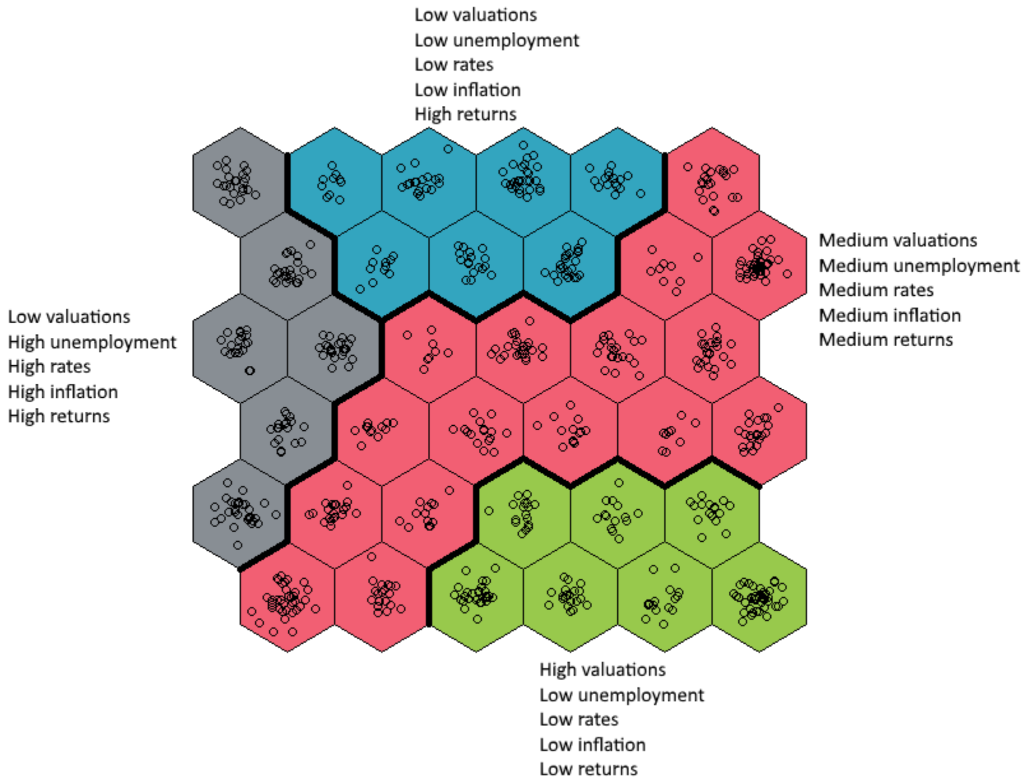
4. Training SOM







4. Training SOM



Result : close grid = similar characteristics.

4. Training SOM

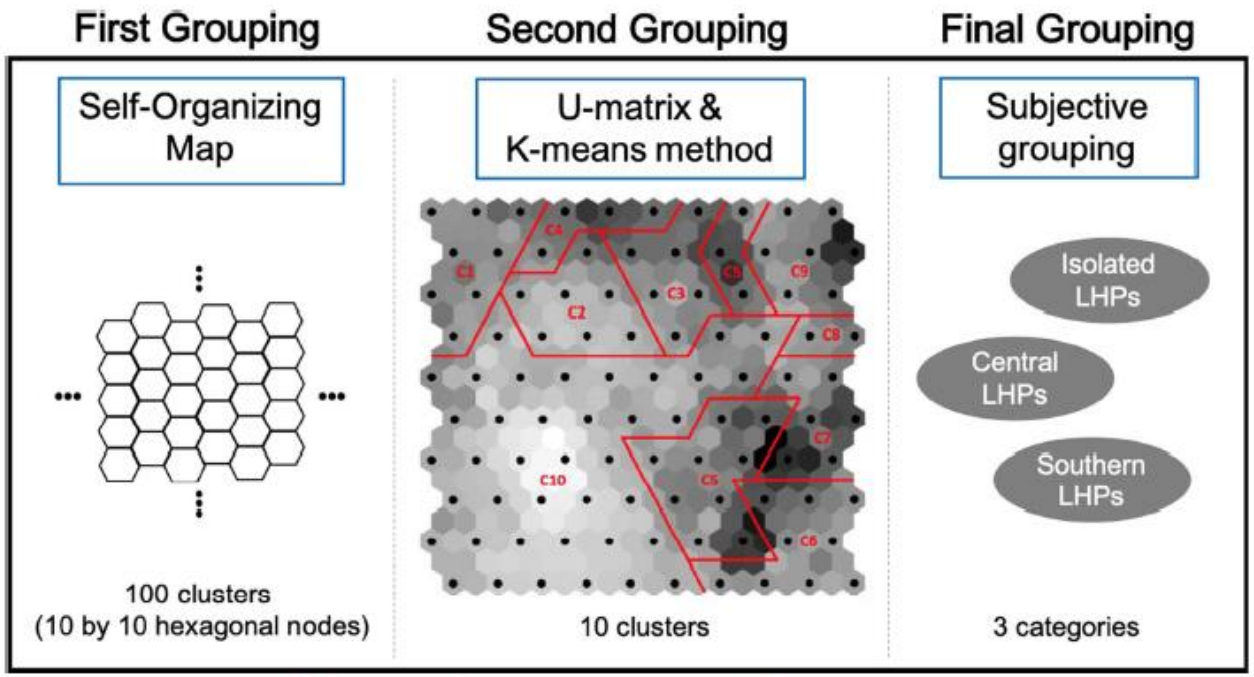


Fig. 1 Schematic diagram representing the three main clustering steps. First, the SOM method is used in classifying rainfall patterns of LHREs with a large 10 × 10 array of nodes. Second, the 100 nodes are narrowed down to 10 clusters using the K-means and U-matrix methods. Finally, the 10 clusters are grouped into three categories based on the regional characteristics of LHREs

Jo, Enoch, et al. "Classification of localized heavy rainfall events in South Korea." *Asia-Pacific Journal of Atmospheric Sciences* 56.1 (2020): 77-88.

5. Python Code for SOM

Not in sklearn…