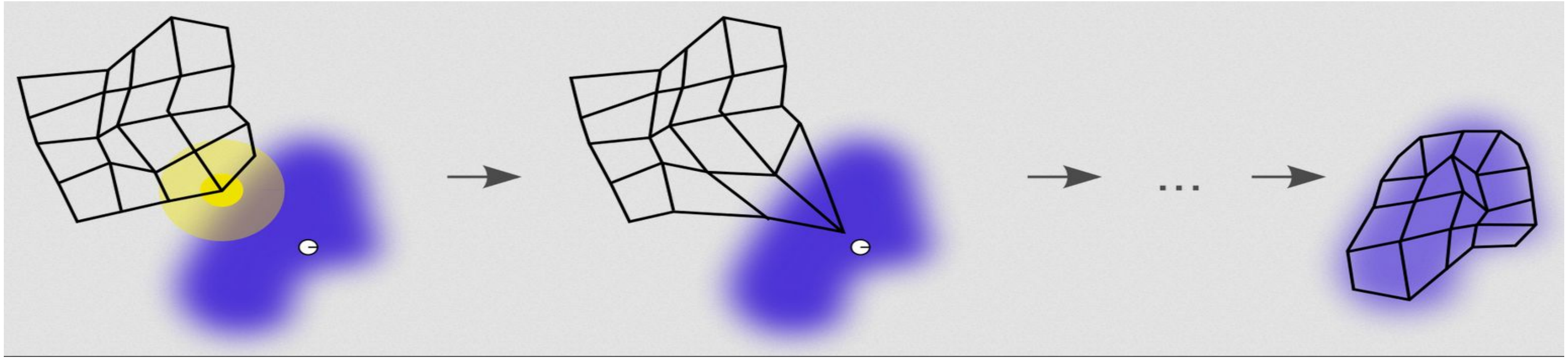
Unsupervised classification using self organising maps

Author: Giriraj Pawar

Index

An illustration of the training of a self-organizing map



Reference:https://en.wikipedia.org/wiki/Self-organizing_map#/media/File:Somtraining.svg

- The blue area is the distribution of the training data, and the small white circle is the current training sample.
- At first (left) the SOM nodes are randomly positioned in the data space.
- The node (highlighted in yellow) which is nearest to the training sample is selected, is called **BMU**.
- In the process of training, the **BMU**(**Best Matching Unit**) node is moved towards the training sample, as well as its neighboring nodes also moved towards training sample. After many iterations the grid tends to approximate the data distribution (right).

Steps in Self-organizing map algorithm

- Initialization.
- Competition.
- Cooperation.
- Adaptive process.

Definitions

- **Initialization:** All the weights are initialized with random values.

- **Competition:** Each neuron competes **discriminant function** which provides the basis for competition. The particular neuron with the highest value of the discriminant function is declared the winner **BMU (Best Matching Unit)**. (**Smallest euclidean distance between the input vector and the weight vector of respective neuron**).

- **Cooperation:** The winning neuron determines the spatial location of a topological neighbourhood of excited neurons, thereby providing the basis for cooperation among neighbouring neurons.

- **Adaptive process:** The excited neurons increase their individual values of the discriminant function in relation to the input pattern through **weights adjustment,** so that the response of the winning neuron to the subsequent application of a similar input pattern is enhanced.

Competition

$X \ = [X_1, X_2, X_3, \ldots\ldots X_m]^T$

$W_j = [W_{j1}, W_{j2}, W_{j3} \ldots\ldots W_{jm}]^T$

m dimensional input.

We have to find best match between X and $W_{j.}$

Compute $\mathbf{W_j^T X}$ for j = 1,2,3,........,l.

And select the largest among this **maximizing** $\mathbf{W_j^T X}$ and **minimizing** $\| \mathbf{X - W_j} \|$.

So index **i(X)** of winning neuron and corresponding closest weight vector can be given by.

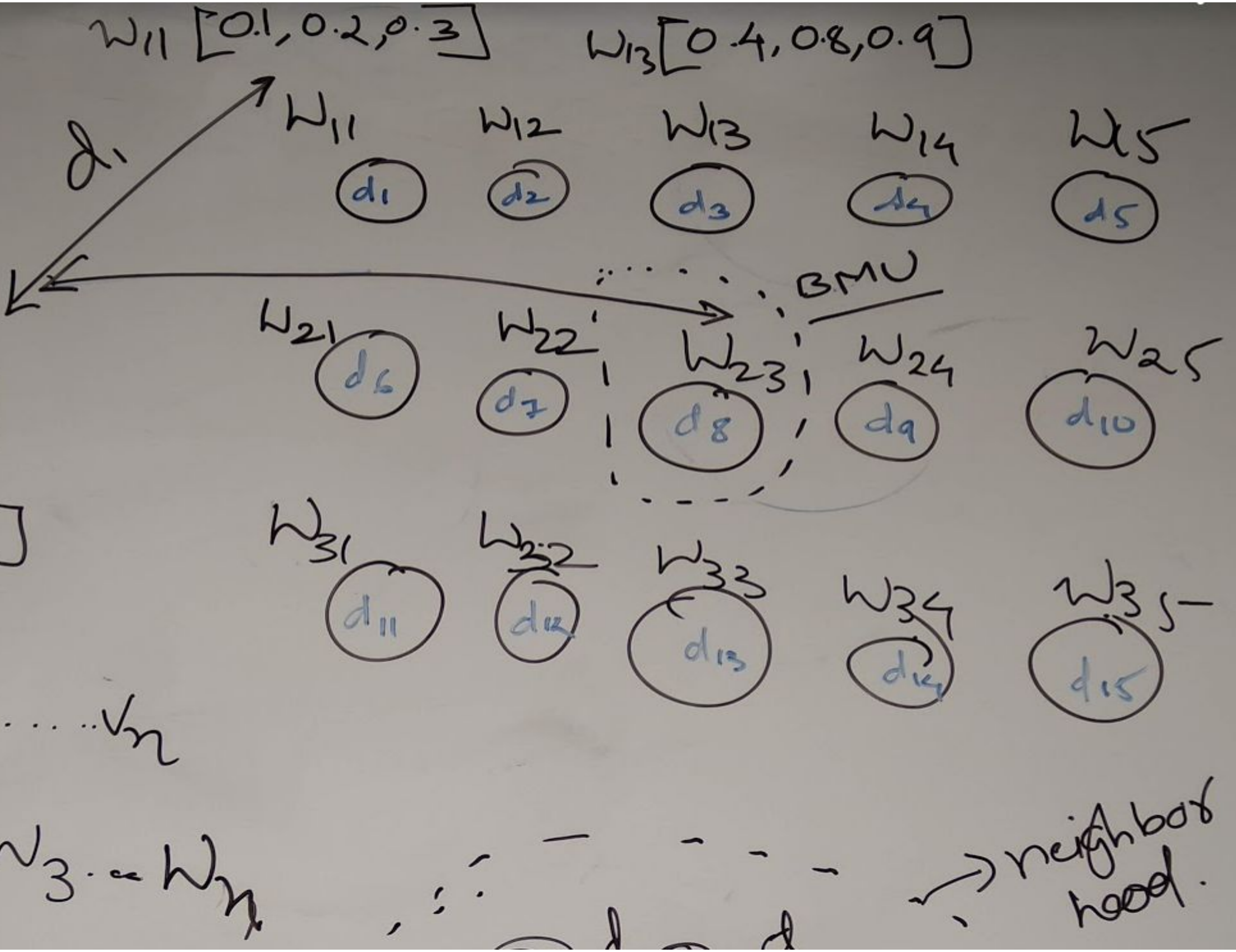
$$\mathbf{i(X) \ = \ arg\ min_j \| X_i - W_j \| \ \text{-----}\ (1)}$$

INPUT

$$\begin{array}{c|ccc} & R & G & B \\ v_1 & 100 & 4 & 5 \\ v_2 & 3 & 100 & 5 \\ v_3 & 2 & 4 & 110 \end{array}$$

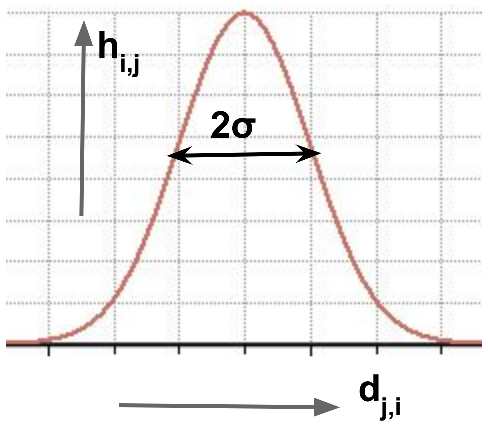
Input vectors $\rightarrow V_1, V_2, V_3 \ldots\ldots V_n$

weights $\rightarrow W_1, W_2, W_3 \ldots W_n$



Cooperation

- It is a mechanism of finding relationship between winning neuron and its surroundings.
- When one neuron fired, its closest neighbours tend to get excited more than those further away.
- Intuitively, if we go further away from the winning neuron, the neighborhood function gradually decreases.
- Neighborhood function is always maximum at the winning neuron and it should decrease monotonically when it is moves away from the winning neuron.



We can use gaussian function as neighborhood function. It monotonically decaying function with distance $d_{j,i}$.

continue.....

If winning neuron is **i**

Then **topological neighborhood** can be given by $\mathbf{h_{j,i}}$ which is centered around **i** and encompassing neuron **j.**

$\mathbf{d_{j,i}}$ lateral distance between the winning neuron **i** and excited neuron **j.**

$$\mathbf{h_{j,i} = \exp\left(- d_{j,i}^2 \;/\; 2\sigma^2 \right) \text{ ----- } (2)}$$

$$\mathbf{d_{j,i}^2 = \| r_j - r_i \|^2}$$

$\mathbf{r_j}$ = Position vector of the excited neuron.

$\mathbf{r_i}$ = Position vector of the winning neuron.

$\boldsymbol{\sigma}$ = width of gaussian function, size of neighborhood, as Iterations increases σ value decreases with time.

continue.....

σ is not constant with respective time as number of iterations increases σ value decreases, hence In SOM learning algorithm the area of the neighbourhood shrinks over time.

$\sigma(n) = \sigma_o \exp(-n / T_1)$ exponential decay function.

T_1 is time constant.

σ_o is the initial σ.

n = 0, 1, 2, ………… (iterations).

$h_{j,i(X)}(n) = \exp(-d_{j,i(X)}^2 / 2\sigma^2(n))$ ----- (3) This equation is called neighborhood function.

Adaptive process or process of learning

- SOM uses Hebbian learning.
- In this process the winning neuron and its neighbours will have their weights updated.
- The newly adjusted weight for the node is equal to the old weight **W**, plus a fraction of the difference η between the old weight **W** and the input vector **X**.
- The update formula for a neuron j with weight vector W_j is give by below equation.

$$\mathbf{W_j(n+1) = W_j(n) + \eta(n) \, . \, h_{j,i(X)}(n) \, . \, (X - W_j)}$$

Above equation is called **topological ordering**.

$\boldsymbol{\eta}$**(n)** $= \eta_o$ **exp(-n / T2)** is learning rate.

T2 is time constant and n = 0,1,2,........

continue....

There are two phases in adaptive process:

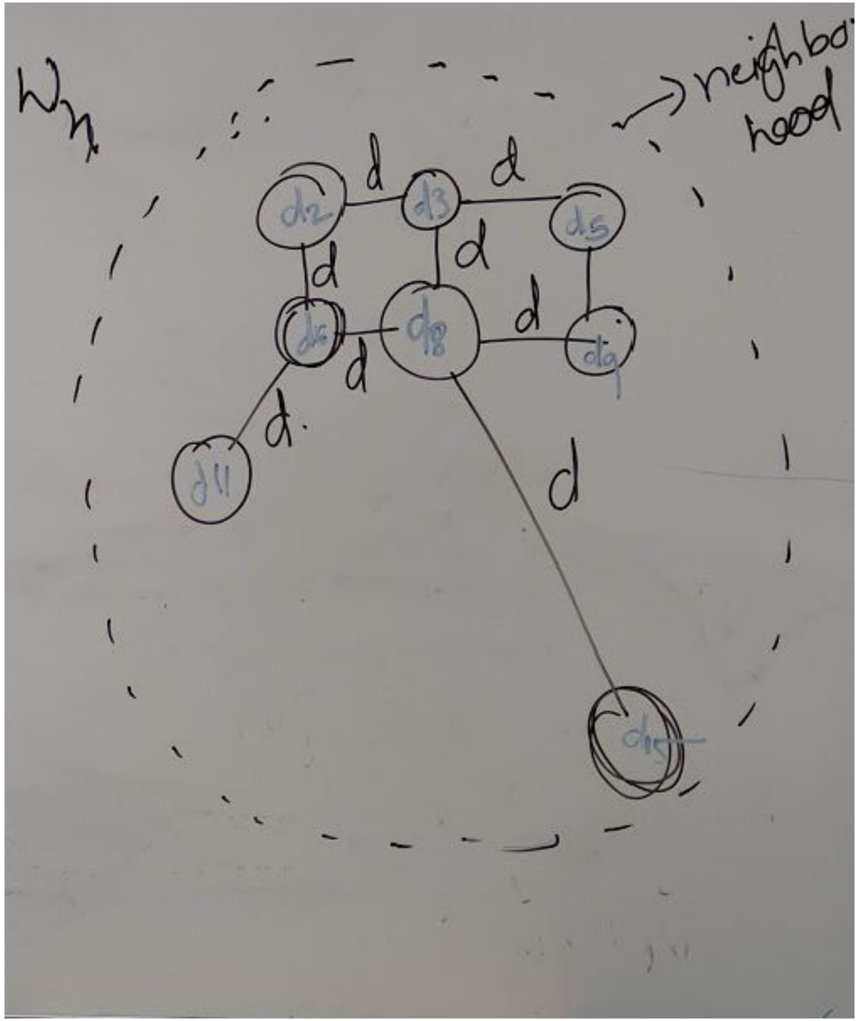
- **Ordering or self-organizing phase**: In this process, the **topological ordering** of nodes takes place. This will take **1000 iterations**, and we have to careful while choosing various parameters like learning rate.

- **Convergence phase:** In this process, the positions of the nodes that are **obtained** by the **topological ordering** phase are **fine tuned**, to **reduce loss and increase performance**. The number of iterations in this phase will be **at least 500 times the number of neurons** in the map, parameters must be chosen carefully.

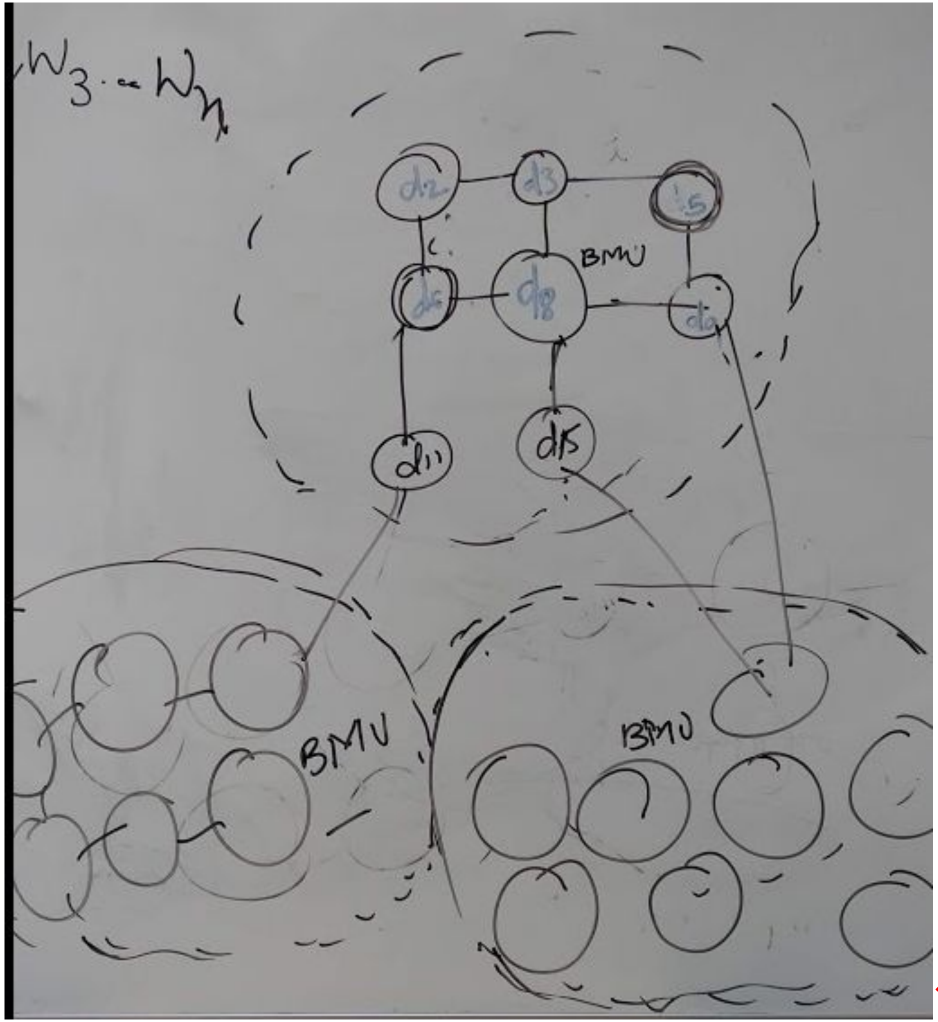
- Topology size increases number of **iterations in convergence phase** will increase.

We have to provides number of nodes as an input to create the random topological map.

Ex: 10 x 10 map will be having 100 nodes.



W_n
neighbor hood
d2
d3
d5
d6
d8
d9
d11
d15
d



$W_3 \dots W_n$
d2
d3
5
d6
d8
d9
d11
d15
BMU
BMU
BMU

Algorithm [1]

1. Randomize the node weight vectors in a map.
2. Randomly pick an input vector.
3. Traverse each node in the map.
    1. Use the Euclidean distance formula to find the similarity between the input vector and the map's node's weight vector.
    2. Find the node that produces the smallest distance (this node is the best matching unit, **BMU**).
4. Update the **weight vectors of the nodes in the neighborhood** of the **BMU** (**including the BMU itself**) by making them closer to the input vector.
5. Repeat from step 2.

References

[1] “Self-organizing map - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Self-organizing_map. [Accessed: 24-Jan-2020].

[2] “(2) Lec-35 Introduction to Self Organizing Maps - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=LjJeT7rwvF4. [Accessed: 26-Jan-2020].