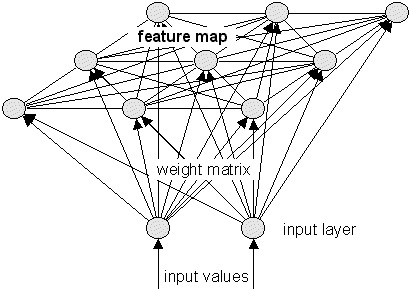
**Kohonen SOM**

**What are Kohonen's networks**?

Kohonen's networks are one of basic types of self-organizing neural networks found by [Teuvo Kohonen](https://en.wikipedia.org/wiki/Teuvo_Kohonen) in the 1980s. It’s known as self-organizing since the network changes its weights according to the input without the intervention of a supervisor. During biological neural activation via axons, Cells close to the active cell have excitatory links whose strength drops as the distance increases. The Kohonen neural network is built on similar lines by restricting the adjustment of weight values to near "neighborhoods".

**Architecture of the Kohonen Network**



The Kohonen network consists of an input layer and a grid or a feature map which consists of neurons or records randomly selected from the input .Each neuron or record in the grid of feature map is connected to other neurons in its neighborhoods and weight updates is restricted thru these neighborhoods.

Grid or feature map size is determined by sqrt (5\*dlen^0.54321) .This recommendation is taken from som tool box. <http://www.cis.hut.fi/somtoolbox/package/docs2/som_make.html>

**Core Kohonen Clustering Algorithm:**

**Inputs**:

1. Scaled input data as a matrix
2. Somgrid: Grid size determined by  sort(5\*dlen^0.54321) <http://www.cis.hut.fi/somtoolbox/package/docs2/som_make.html>
3. rlen: the number of times the complete data set will be presented to the network . During som training , rlen\*number of records iterations happen
4. Alpha: learning rate, a vector of two numbers indicating the amount of change. Default is to decline linearly from 0.05 to 0.01 over rlen updates

**Core Algorithm**

1. Random sample from i/p data is chosen whose count matches the number of neurons selected in the grid . If the grid chosen is 2\*2 then 4 records from the input dataset is chosen randomly
2. A record is chosen randomly from the input and the Euclidean distance is calculated with each of the records chosen as part of the grid.

Distance =

I = randomly selected record from the i/p dataset

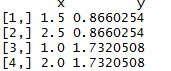
W = record selected from the grid

n = number of columns

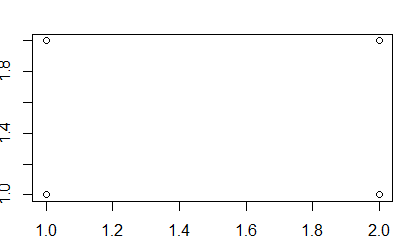
1. The record from the grid whose Euclidean distance with respect to the randomly chosen record from the grid is the shortest is chosen as the Best matching Unit.
2. The grid is initialized with the following x and y parameters according to the formulae
3. Chosen X and Y values are expanded. for the grid size 2\*2 ,



1. The coordinate points are expanded as a matrix in the following form



These points can be visualized as



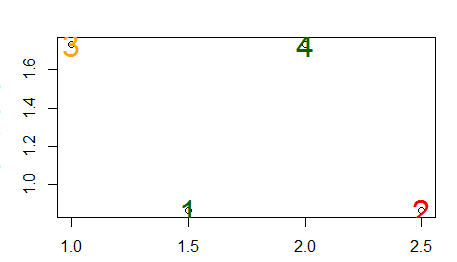
1. These points are modified to form a hexagonal grid structure of the format

as

x <- x + 0.5 \* y%%/2

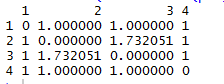
y <- sqrt(3)/2 \* y

Visualizing these points on the grid:



1. The distance between the coordinates are calculated as

Distance= as.matrix(stats::dist(pts))



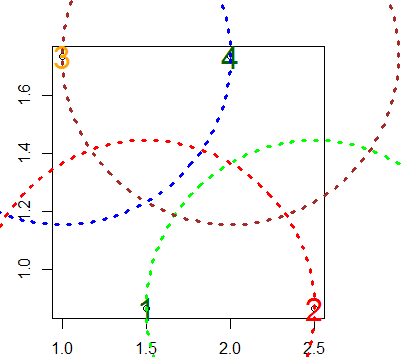
1. Once the distance is calculated , the starting radius of these neurons are calculated to cover 2/3 0r 67% percent of the unit distances i.e



This comes to +1 -1 .

Visualizing this

1. Neuron 1 has its radius 🡪 Red
2. Neuron 2 has its radius 🡪 Green
3. Neuron 3 has its radius 🡪 Blue
4. Neuron 4 has its radius 🡪 Brown



Those records which lie within the radius of Best matching unit is updated depending upon the learning rate where as those beyond the radius of the BMU is not updated. The threshold radius is determined as

* 1. Threshold \_radius= radii [0] - (radii [0] - radii [1]) \* (Current iteration/Total number of iterations)
  2. In R, default value for radii [0] =1 and radii [1] =-1
  3. Thus with each iteration those records within the radius of the winning record is shrunk accordingly
  4. Update Rule:

If the radius of the record on the grid < = threshold\_radius

=+

Else

=

Where New Weights

= Original Weights

Change in weights

E.g.: consider the case i.e. after 1st iteration, suppose record 2 i.e. 3nd neuron is the BMU.Then any other neuron outside of the radius of the winning neuron does not get updated with new weights.

Based on unit distances between the points on the grid (Refer para 5),

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 |
| 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1.7 | 1 |
| 2 | 1 | 1.7 | 0 | 1 |
| 3 | 1 | 1 | 1 | 0 |

Distance between (2,0) ->1

Distance between (2,1) ->1.7

Distance between (2,2)->0

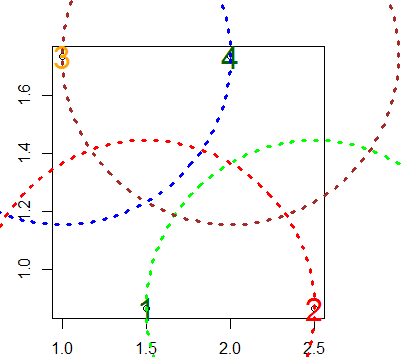
Distance between (2,3)->1

Since the starting radius is 1 for the winning BMU, Anything within the radius has its weights updated where as anything outside this radius does not have its weight updated.

Only 2nd neuron ( ie neuron 1 as starting from 0) is outside the radius of BMU ( i.e. 1.75 >1 )

Thus the updated weights are as follows. The 2nd neuron is not updated and highlighted in yellow.

1. Neuron 1 has its radius 🡪 Red
2. Neuron 2 has its radius 🡪 Green
3. Neuron 3 has its radius 🡪 Blue
4. Neuron 4 has its radius 🡪 Brown



Thus as radius of neuron 3 (Winning BMU) does not overlap with that of neuron 2 or record1 weights don’t fall in the neighborhood and hence don’t get updated.

1. The learning rate is shrunk according to formulae:

=+ )

Where = New learning rate

= previous learning rate

1. The following above process is repeated according to the number of iterations.
2. Post iterations , number of clusters are chosen either using a scree plot of hierarchical clustering using dendogram
3. Final clustering on the updated grid is performed by hybrid hierarchical k means

* Perform hierarchical clustering and cut the grid into chosen number of clusters
* Compute the centers of clusters defined by hierarchical clustering. Cluster centers are nothing but the means of variables in clusters.
* Having chosen the number of clusters K means is applied on the grid consisting of the records which have been updated during the iterations to give the clusters.
* K-means clustering is performed on hierarchical clustering defined cluster centers