Self Organizing Map

Abhilash Poudel - MECE 16901

Overview

- Introduction
- Algorithm
- Example
- Application
- Conclusion

The problem is how to find out similar relationship among lots of information without manual process; get information about data without knowing where to search for it and without knowing where to put the new data

Introduction [Origin]

- Introduced by C. von der Malsburg(1973), developed and refined by T.Kohonen(1982)
- Neural Network Algorithm using unsupervised competitive learning
- Used for organization and visualization of data
- Neurons are arranged on a flat grid
- No hidden layer, only input and output layer
- Each neuron on the grid is an output neuron

Introduction [Concept]

- Make a 2-D array and randomize it i.e. initialize weight
- Present training data to the map and let the cells on the map compete to win(Usually Euclidean distance is used)
- Simulate the winner i.e. updating weight matrix alongside its neighbour
- Repeat these steps number of times
- The final result is 2-D “weight” array

Introduction

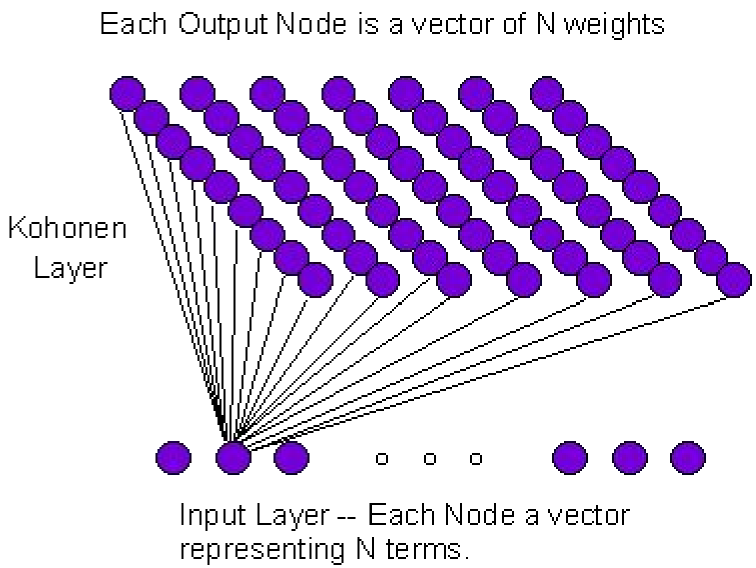


Fig: Kohonen Network

Algorithm

1. Initialize weights. Set max value of R, set learning rate α
2. While stop condition is false repeat 3 to 8
3. For each input vector **x** do steps 4 to 6
4. For each **j** neuron, compute the Euclidean distance

$$D(j) = \sqrt{\sum_{i=1}^{n} (x_i - w_{ij})^2}$$

5. Find the index **j** such that D(j) is minimum

Algorithm

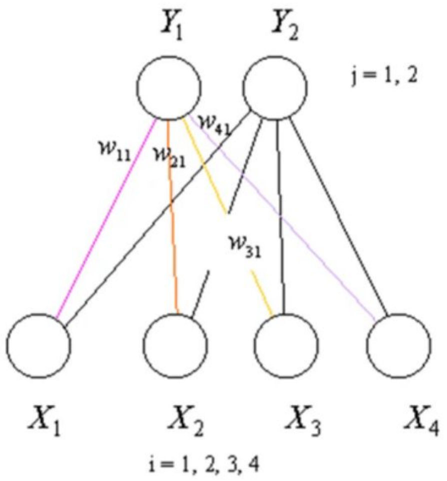
6. For all the neurons j, update the weight to the new one alongside with neighbour

$$w_{ij}(new) = w_{ij}(old) + \alpha(x_i - w_{ij}(old))$$

7. Update learning rate α. It is a decreasing function to limit the number of iterations
8. Test stop condition. Typically α so small that it is insignificant with weight update

Example[Initialize]

To make this simple, let's take a simple example with 2 neurons at output layer as shown below



Let Initial weight matrix be

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{bmatrix}$$

Example[Initialize]

x1	x2	x3	x4
1	1	0	0
0	0	0	1
1	0	0	0
0	0	1	1

Let there be 4 input training patterns

Initial alpha be α = 0.6

and learning rate be defined as $\alpha(t+1) = \frac{\alpha(t)}{2}$

Let topological radius R = 0

x1	x2	x3	x4
1	1	0	0
0	0	0	1
1	0	0	0
0	0	1	1

Example[Sample]

For vector **1100**(We are using Euclidean distance squared for easy mathematics)

$D(1) = (1-0.2)^2+(1-0.6)^2+(0-0.5)^2+(0-0.9)^2 = 1.86$

$\mathbf{D(2) = (1-0.8)^2+(1-0.4)^2+(0-0.7)^2+(0-0.3)^2 = 0.98}$

Hence j = 2. Since R = 0, we don't consider neighbours, so we update weight for neurons.

x1	x2	x3	x4
1	1	0	0
0	0	0	1
1	0	0	0
0	0	1	1

Example[Update]

And the new updated weight matrix becomes as

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{bmatrix}$$

$w_{12}(new) = w_{12}(old) + \alpha(x_1 - w_{12}(old))$
$= 0.8 + 0.6(1-0.8) = 0.92$

x1	x2	x3	x4
1	1	0	0
0	0	0	1
1	0	0	0
0	0	1	1

Example[Sample & Update]

For vector **0001**

D(1) = 0.66, D(2) = 2.2768 Hence j = 1

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.92 \\ 0.24 & 0.76 \\ 0.20 & 0.28 \\ 0.96 & 0.12 \end{bmatrix}$$

x1	x2	x3	x4
1	1	0	0
0	0	0	1
1	0	0	0
0	0	1	1

Example[Sample & Update]

For vector **1000**

D(1) = 1.8656, **D(2) = 0.6768** Hence j = 2

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.968 \\ 0.24 & 0.304 \\ 0.20 & 0.112 \\ 0.96 & 0.048 \end{bmatrix}$$

x1	x2	x3	x4
1	1	0	0
0	0	0	1
1	0	0	0
0	0	1	1

Example[Sample & Update]

For vector **0011**

D(1) = 0.7056, D(2) = 2.724 Hence j = 1

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.032 & 0.968 \\ 0.096 & 0.304 \\ 0.680 & 0.112 \\ 0.984 & 0.048 \end{bmatrix}$$

Example[Final Stage]

Now we reduce the learning rate to $\alpha(1) = \frac{\alpha(0)}{2} = \frac{0.6}{2} = 0.3$

After **100 iterations** for all input vectors, the final weight matrix becomes

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 6.7 \times 10^{-17} & 1 \\ 2 \times 10^{-16} & 0.49 \\ 0.51 & 2.3 \times 10^{-16} \\ 1 & 1 \times 10^{-16} \end{bmatrix}$$

The matrix rounding looks like

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0.5 \\ 0.5 & 0 \\ 1 & 0 \end{bmatrix}$$

Cluster 1

Cluster 2

Example[Grouping]

After we have find the weight, we calculate the cluster/group for each inputs with same Euclidean distance formula

x1	x2	x3	x4	Cluster
1	1	0	0	2
0	0	0	1	1
1	0	0	0	2
0	0	1	1	1

Example[Testing]

Let us take a sample data **1010** and try to find out which cluster do it belong

$D(1) = (1-0)^2+(0-0)^2+(1-0.5)^2+(0-1)^2 = 2.25$

$\mathbf{D(2) = (1-1)^2+(0-0.5)^2+(1-0)^2+(0-0)^2 = 1.25}$

Hence j=2 so, this test data must lie in **Cluster 2**

DEMO

Application

- SOM to cluster data of IRIS Dataset
  - It is a flower dataset consisting 50 samples from 3 species of Iris
- Can be applied in DNA categorization
- Can be used in OCR by clustering and verifying based on the cluster

Conclusion

- SOM projects high-dimensional data to a 2D map
- Projection preserves topology of data so similar data maps to nearby locations on map
- SOM has many practical applications in pattern recognition, speech analysis, industrial and medical diagnostics, data mining
- Large quantity of good quality representative training data required
- Doesn’t give output but cluster/group inputs

References

- http://www.ai-junkie.com/ann/som/som1.html
- [Self Organizing Map (SOM) tutorial]

  https://www.youtube.com/watch?v=abF_FdCb5OI
- http://mnemstudio.org/neural-networks-som3.htm

THANKS!

Any questions?