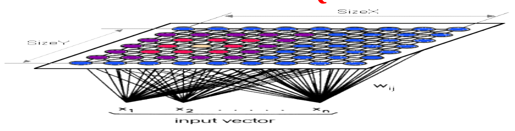


Neural Networks and Fuzzy Logic (630514)

Lecture 16

Self-organizing map using matlab

Create a Self-Organizing Map Neural Network: selforgmap

Syntax:

selforgmap (dimensions, coverSteps, initNeighbor, topologyFcn, distanceFcn)

takes these arguments:

dimensions	Row vector of dimension sizes (**default = [8 8]**)
coverSteps	Number of training steps **(default = 100)**
initNeighbor	Initial neighborhood size **(default = 3)**
topologyFcn	Layer topology function **(default = 'hextop')**
distanceFcn	Neuron distance function **(default = 'linkdist')**

and returns a **self-organizing map**.

- The neurons in the layer of an **SOFM** are arranged originally in physical positions according to a **topology function**. The function **gridtop**, **hextop**, or **randtop** can arrange the neurons in a grid, hexagonal, or random topology.
- Distances between neurons are calculated from their positions with a **distance function**. There are four distance functions, **dist, boxdist, linkdist,** and **mandist**. Link distance is the most common.

Examples

Example 1: simplecluster_dataset

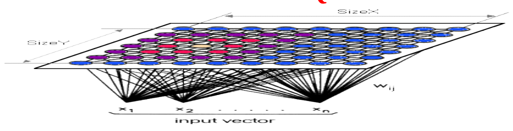
- LOAD simplecluster_dataset.MAT loads these two variables:

simpleclusterInputs - a 2x1000 matrix of 1000 two-element vectors.

[X, T] = simplecluster_dataset loads the inputs and targets into variables of your own choosing.

design an 8x8 clustering neural network with this data at the command line

```
% Clustering Problem using a Self-Organizing Map
load simplecluster_dataset;
x = simpleclusterInputs;
% Create a Self-Organizing Map
dim1 = 10;
dim2 = 10;
net = selforgmap([dim1 dim2]);
% Train the Network
[net,tr] = train(net,x);
% Test the Network
y = net(x);
% View the Network
view(net)
% Plots
figure, plotsomtop(net)
figure, plotsomnc(net)
```



```
figure, plotsomnd(net)
figure, plotsomplanes(net)
figure, plotsomhits(net,x)
figure, plotsompos(net,x)
```

Example 2: **iris_dataset:** This dataset can be used to create a neural network that classifies iris flowers into three types.
LOAD iris_dataset.MAT loads these two variables:
irisInputs - a 4x150 matrix of four attributes of 1000 flowers.

1. ***Sepal length in cm***
2. ***Sepal width in cm***
3. ***Petal length in cm***
4. ***Petal width in cm***

irisTargets - a 3x150 matrix of 1000 associated class vectors defining which of four classes each input is assigned to. Classes are represented by a 1 in one of four rows, with zeros in the others.
[X,T] = iris_dataset loads the inputs and targets into variables of your own choosing.

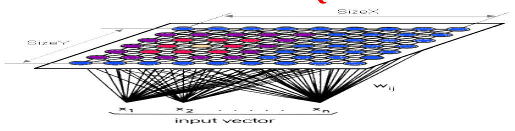
- Here is how to design a **pattern recognition feedforward neural network** with this data at the command line.

```
[x,t] = iris_dataset;
net = patternnet(10);
net = train(net,x,t);
view(net)
y = net(x);
```

- Clustering is the process of training a neural network on patterns so that the network comes up with its own classifications according to pattern similarity and relative topology. This is useful for gaining insight into data, or simplifying it before further processing.

Matlab code for Example 2

```
% Clustering Problem using a Self-Organizing Map
% iris_dataset.
load iris_dataset;
x = irisInputs;
% Create a Self-Organizing Map
dim1 = 10;
dim2 = 10;
net = selforgmap ([dim1 dim2]);
% Train the Network
[net,tr] = train(net,x);
% Test the Network
y = net(x);
% View the Network
view(net)
% Different Plots
figure, plotsomtop(net)
```



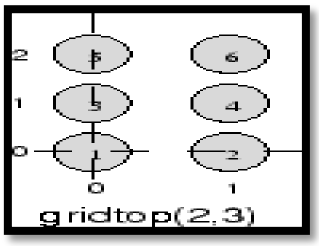
```
figure, plotsomnc(net)
figure, plotsomnd(net)
figure, plotsomplanes(net)
figure, plotsomhits(net,x)
figure, plotsompos(net,x)
```

Topology function:

- The **gridtop** topology starts with neurons in a rectangular grid. For example, suppose that you want a 2-by-3 array of six neurons. You can get this with

```
pos = gridtop(2,3)
   pos =
   0 1 0 1 0 1
   0 0 1 1 2 2
```

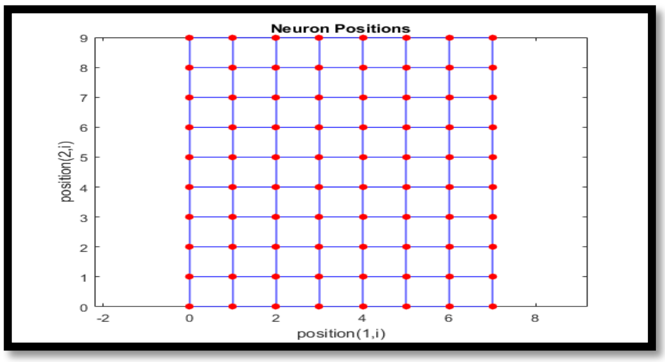
  - Here neuron **1** has the position **(0,0)**, neuron **2** has the position **(1,0)**, and neuron **3** has the position **(0,1)**, etc.



```
pos = gridtop(3,2); % slightly different arrangement
   pos =
   0 1 2 0 1 2
   0 0 0 1 1 1
```

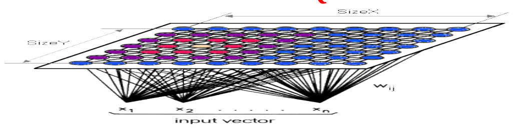
- You can create an 8-by-10 set of neurons in a **gridtop** topology with the following code:

```
pos = gridtop(8,10);
plotsom(pos)
```



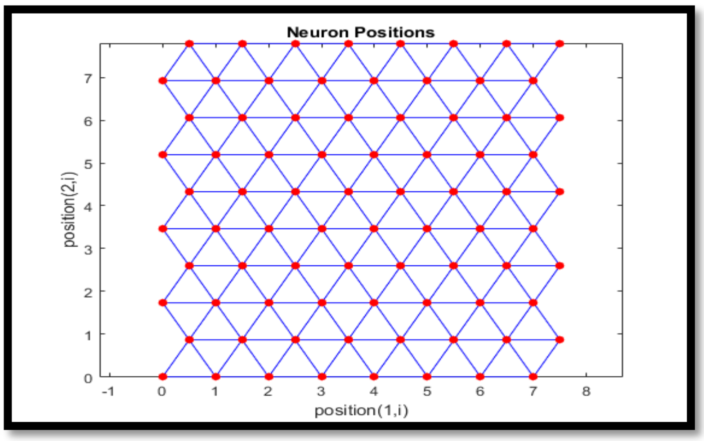
- The **hextop** function creates a similar set of neurons, but they are in a hexagonal pattern. A 2-by-3 pattern of **hextop** neurons is generated as follows:

```
pos = hextop(2,3)
    pos =
    0 1.0000 0.5000 1.5000 0 1.0000
    0 0 0.8660 0.8660 1.7321 1.7321
```



- You can create and plot an 8-by-10 set of neurons in a **hextop** topology with the following code:

  pos = hextop(8,10);
  plotsom(pos)



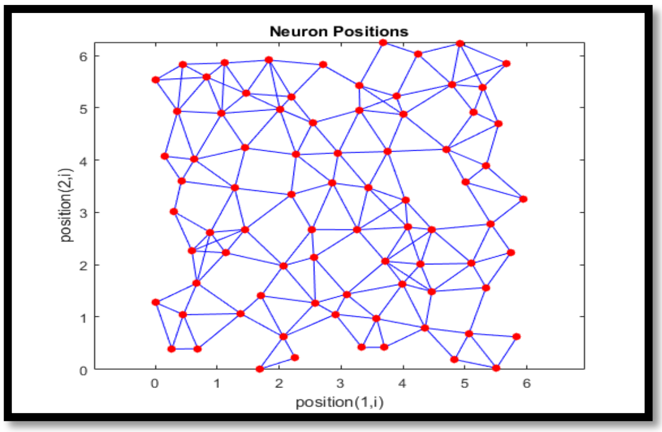
- Note the positions of the neurons in a hexagonal arrangement.
- Finally, the **randtop** function creates neurons in an N-dimensional random pattern. The following code generates a random pattern of neurons.

  pos = randtop(2,3)

```
pos =
0 0.7620 0.6268 1.4218 0.0663 0.7862
0.0925 0 0.4984 0.6007 1.1222 1.4228
```

- You can create and plot an 8-by-10 set of neurons in a randtop topology with the following code:

  pos = randtop(8,10);
  plotsom(pos)



Distance Functions:

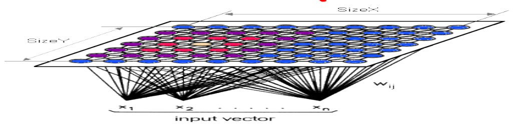
- The **dist** function calculates the Euclidean distance from a home neuron to any other neuron. Suppose you have three neurons:

  pos2 = [0 1 2; 0 1 2]

```
pos2 =
0 1 2
0 1 2
```

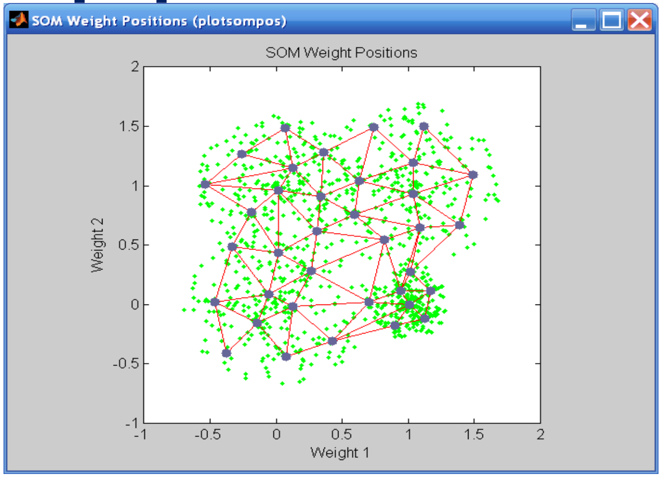
  - You find the distance from each neuron to the other with

  D2 = dist (pos2)

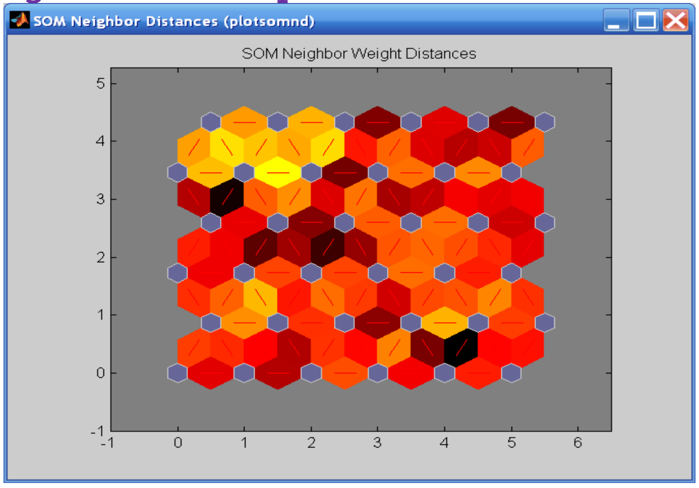


```
D2 =
0              1.4142          2.8284
1.4142         0               1.4142
2.8284         1.4142          0
```

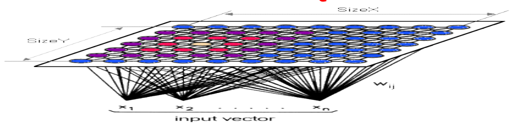
- Thus, the distance from neuron **1** to itself is **0**, the distance from neuron **1** to neuron **2** is **1.414**, etc.
- There are several useful visualizations windows that you can access:
  - **SOM Weight Positions (plotsompos)** button shows the locations of the data points and the weight vectors. As the figure indicates, after only 200 iterations of the batch algorithm, the map is well distributed through the input space.



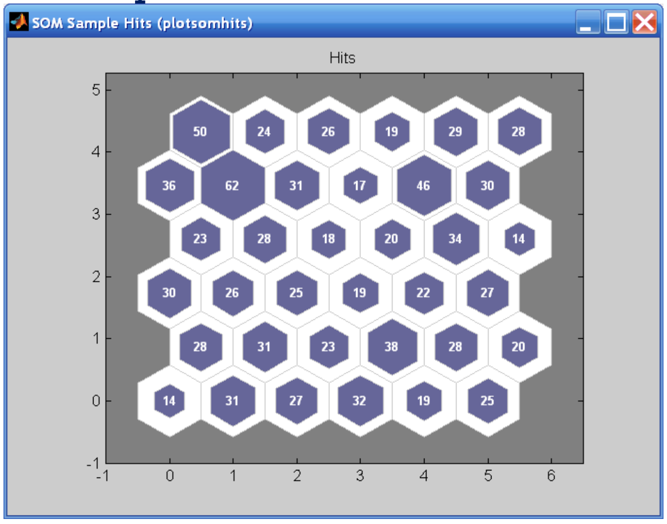
  - **SOM Neighbor Distances (plotsomnd)**. The following figure appears which indicates the distances between neighboring neurons. This figure uses the following color coding:
    - The blue hexagons represent the neurons.
    - The red lines connect neighboring neurons.
    - The colors in the regions containing the red lines indicate the distances between neurons.
    - The darker colors represent larger distances.
    - The lighter colors represent smaller distances.



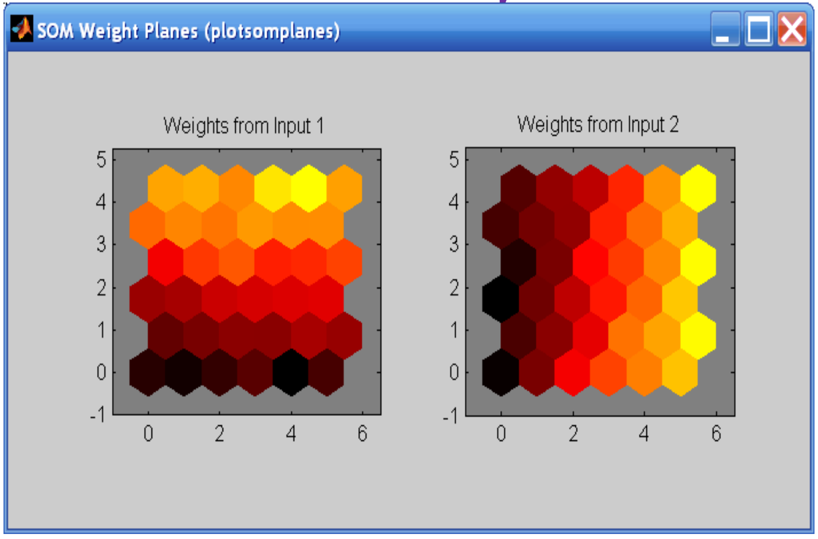
    - A group of light segments appear in the upper-left region, bounded by some darker segments. This grouping indicates that the network has clustered the data into two groups.



- **SOM Sample Hits (plotsomhits)**: **Hit histogram**: visualization way to plot a hit histogram: a good way to view the **distribution** of the data se on the map.



- **SOM Weight Planes (plotsomplanes)** :A **component plane** shows the values in each map unit for one variable (default).
  - Look at differences regarding just one of the input "variables" at a time. From this can be drawn conclusions about which components are the most significant for the classification.
  - Lighter and darker colors represent larger and smaller weights, respectively.
  - If the connection patterns of two inputs are very similar, you can assume that the inputs were highly correlated. In this case, input 1 has connections that are very different than those of input 2.


SOM Weight Planes (plotsomplanes)
Weights from Input 1
Weights from Input 2