SELF ORGANIZING NETWORKS

Self-organized clustering may be defined as a mapping through which N-dimensional pattern spaces are mapped into a smaller number of points in an output space. The mapping is achieved autonomously by a system without supervision, i.e., the clustering is achieved in a self-organizing manner.

In a self-organizing network, the weights of the neurons represent a class of pattern. Input patterns are presented to all the neurons and each neuron produces an output. The value of each neuron's output is used as a measure of how close are the input and the stored patterns (neuron's weights). A competitive learning strategy is used to select the neuron whose weight vector matches closely with the input vector.

Self-organizing refers to the ability to learn and organize information without being given correct answers for the input pattern (unsupervised learning).

FIXED WEIGHT COMPETITIVE NETS

They are additional structures included in networks of multi output in order to force their output layers to make a decision as to which one neuron will fire. This mechanism is called competition. When competition is complete, only one output neuron has nonzero output. Symmetric (fixed) weight nets are: (*Maxnet, Mexican Hat Net and Humming Net*).

1- Maxnet

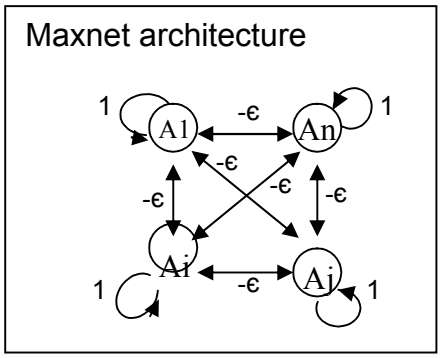
- -Maxnet is based on winner-take-all policy.
- -The n-nodes of Maxnet are completely connected.

- -There is no need for training the network, since the weights are fixed.
- -The Maxnet operates as a recurrent recall network that operates in an auxiliary mode.

Activation functions

$$f(net) = \begin{cases} net & \text{if } net > 0 \\ 0 & \text{otherwise} \end{cases}$$

Where Є is usually positive less than 1 number.



Maxnet Algorithm

Step 1: Set activations and weights,

aj (0) is the starting input value to node *Aj*

$$\omega ij = \begin{cases} 1 & \text{for } i = j \\ -\epsilon & i \neq j \end{cases}$$

Step 2: If more than one node has nonzero output, do step 3 to 5.

Step 3: Update the activation (output) at each node for

$j = 1, 2, 3\ldots\ldots, n$

$aj\ (t+1) = f\ [\ aj\ (t) - \epsilon \sum ai\ (t)]$ $\quad i \neq j$

$\epsilon < 1/m$ where m is the number of competitive neurons

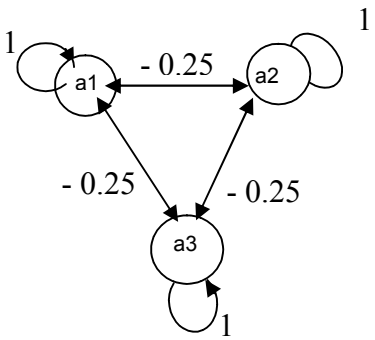
Step 4: Save activations for use in the next iteration.

$aj\ (t+1) \rightarrow aj\ (t)$

Step 5: Test for stopping condition. If more than one node has a nonzero output then Go To step 3, Else Stop.

***Example*:** A Maxnet has three inhibitory weights a 0.25 (ϵ = 0.25). The net is initially activated by the input signals [0.1 0.3 0.9]. The activation function of the neurons is:

$$f(net) = \begin{cases} net & \text{if } net > 0 \\ 0 & \text{otherwise} \end{cases}$$



Find the final winning neuron.

Solution:

First iteration: The net values are:

a**1** (1) = f [0.1 - 0.25(0.3+0.9)] = 0

a**2** (1) = f [0.3 - 0.25(0.1+0.9)] = 0.05

a**3** (1) = f [0.9 - 0.25(0.1+0.3)] = 0.8

Second iteration: a**1** (2) = f [0 - 0.25(0.05+0.8)] = 0

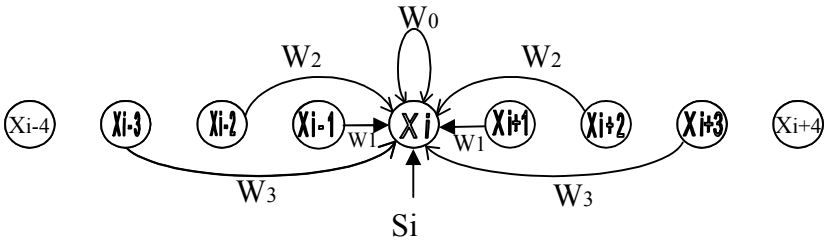
a**2** (2) = f [0.05 - 0.25(0 +0.8)] = 0

a**3** (2) = f [0.8 -0.25(0+0.05)] =0.7875

Then the 3rd neuron is the winner.

2- Mexican Hat Network

- It is more contrast enhancing subnet than the Maxnet.
- Each neuron is connected to other neurons through excitatory and inhibitory links.
- Neurons which are in close proximity are connected through excitatory (positive) links and are called "*cooperative neighbors"*
- Neurons of farther away positions are connected through inhibitory (negative), links and are called "competitive *neighbors*".
- Neurons beyond the competitive neighbors are unconnected.
- Each neuron in a particular layer receives an external signal in addition to the interconnection signals.



Activation function

$$f(\text{net}) = \begin{cases} \text{net} & \text{net} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\boldsymbol{K = i + R}$$

$$Xi = f\,[Si + \sum W\boldsymbol{k}\, X\boldsymbol{k}\,(t-1)\,]$$

$$\boldsymbol{K = i - R}$$

- The interconnections of each neuron have two symmetrical regions around it. The region near the neuron within radius (R1) has positive weights, and the region outside R1 and within R2, where connections are negative.

TERMINOLOGY

R1: Radius of reinforcement region

R2 Radius of connection, (R1 < R2)

W***k***: Weights of Xi neighbors

W***k*** is positive if $0 < k \leq R1$

W***k*** is negative if $R1 < k \leq R2$

W***k*** is zero if $k > R2$

X: Vector of activations

X_old: Vector of activations at previous step

t_max: Total number of iterations of contrast enhancement

Si: External signal

ALGORITHM

Step 1: Set values of R1, R2 and t_max (No. of iterations).

Initialize weights:

W***k*** = C1 for ***k*** = 0, 1, 2, R1, where C1 is random > 0.

W***k*** = C2 for ***k*** = R1+1,R1+2,….,R2 (C2 is random > 0).

Step 2: (t = 0)

Present the external signal vector S: X=S

Save activations in X_old array for i = 1, 2, n, to be used in the next step.

X_old = X = (x1, x2, xn) t= 1

Step 3: Calculate the net input for i =1, 2, n.

(net)**i** = X**i** = C1∑ (x_old)**i+k** – C2 ∑ (x_old)**i+k** – C2 ∑ (x_old) **i+k**

Step 4: Apply activation function and save current activations in X_old to be used in the next iteration.

Step 5: Increment counter t, as t = t +1.

Step 6: Test for stopping condition:

If t < t_max, then continue (Go to step 3), else stop.

***Note**:* The positive and negative reinforcement have the effect of increasing the activation of units with large initial values and reducing that of units with small initial activations respectively.

Example: A Mexican Hat net consists of seven input units. The net is initially activated by the input signal vector [0.0 0.4 0.7 1.0 0.7 0.4 0.0]. The activation function of the neuron is:

$$f(x)= \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \le x \le 2 \end{cases}$$

The max. number of iterations is three.

Solution:

Step 1:

Let R1=1, R2=2, C1=.7, and C2= -0.3 (initialization).

Step 2: (t = 0)

X = S = [0.0 0.4 0.7 1.0 0.7 0.4 0.0]

X_old = [0.0 0.4 0.7 1.0 0.7 0.4 0.0], (presenting external signal vector and saving activations in X_old array).

t = 1

Step 3:

X1 = 0.7(0.0+0.4) – 0.3(0.7) = 0.07

X2 = 0.7(0.0+0.4+0.7) – 0.3(1.0) = 0.47

X3 = 0.7(0.4+0.7+1.0) –0.3(0.0+0.7) = 1.26

X4 = 0.7(0.7+1.0+0.7) – 0.3(0.4+0.4) = 1.44

X5 = 0.7(1.0+0.7+0.4) – 0.3(0.7+0.0) = 1.26

X6 = 0.7(0.7+0.4+0.0) – 0.3(1.0) = 0.47

X7 = 0.7(0.4+0.0) – 0.3(0.7) = 0.07

Step 4:

X_old = [0.07 0.47 1.26 1.44 1.26 0.47 0.07]

Step 5:

t = t+1= 2

Step 6:

t = 2, Go to step 3

Step 3:

X_old = [0.07 0.47 1.26 1.44 1.26 0.47 0.07]

X1 = 0.7(0.07+0.47) – 0.3(1.26) = 0

X2 = 0.7(0.07+0.47+1.26) – 0.3(1.44) = 0.828

X3 = 0.7(0.47+1.26+1.44) – 0.3(0.07+1.26) = 1.82

X4 = 0.7(1.26+1.44+1.26) – 0.3(0.47+0.47) = 2.49

X5 = 0.7(1.44+1.26+0.47) – 0.3(1.26+0.07) = 1.82

X6 = 0.7(1.26+0.47+0.07) – 0.3(1.44) = 0.828

X7 = 0.7(0.47 +0.07) – 0.3(1.26) = 0

Step 4:

X_old = [0.0 0.828 1.82 2.49 1.82 0.828 0.0]

Step 5:

t = t+1= 3, then stop.

The network's outputs are shown for t = 0, 1 and 2.

3- Hamming Net

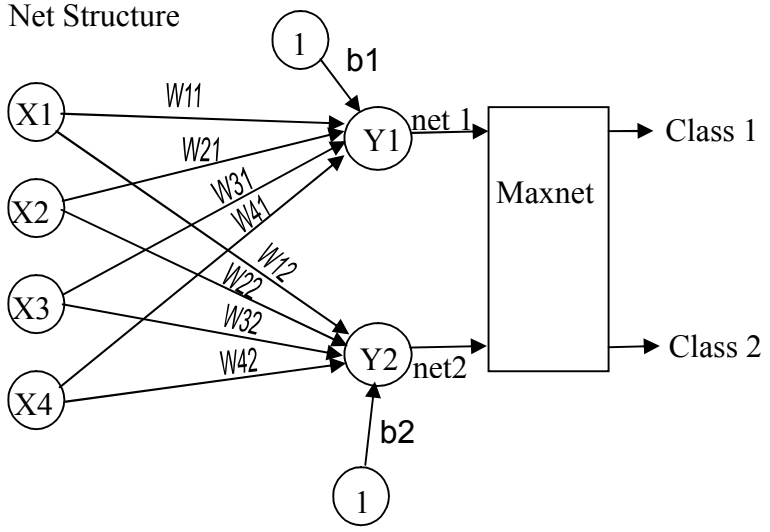
Hamming net is a maximum likelihood classifier net. It is used to determine an exemplar vector which is most similar to an input vector. The measure of similarity is obtained from the formula:

x.y = a – D = 2a – n , since a +D = n

Where D is the hamming distance (number of component in which vectors differ), a is the number components in which the components agree and n is the number of each vector components.

When weight vector off a class unit is set to be one half of the exemplar vector, and bias to be (n/2), the net will find the unit closest exemplar by finding the unit with maximum net input. Maxnet is used for this purpose.

Hamming Net Structure



Wij = ei(j)/2

Where ei(j) is the i'th component of the j'th exemplar vector.

Terminology

M : number of exemplar vectors

N : number of input nodes (input vector components)

E(j) : j'th exemplar vector

Algorithm

Step 1: Initialize the weights

wij = ei(j)/2 = i'th component of the j'th exemplar,

(i= 1,2,....n, and j = 1,2,.....m)

Initialize bias values, bj = n/2

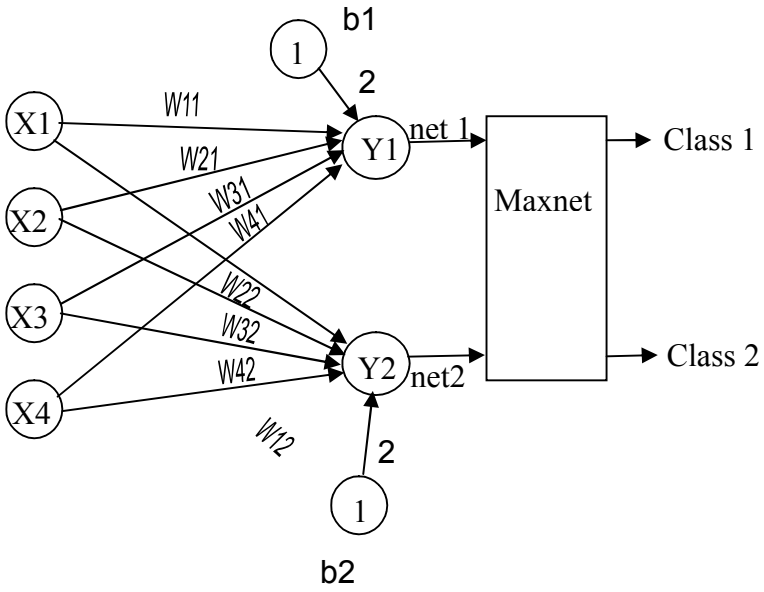
For each input vector X do steps 2 to 4

Step 2: Compute net input to each output unit Yj as:

Yinj = bi + ∑eij xij (i = 1,2,…n, j = 1,2,…m)

Step 3: Maxnet iterations are used to find the best match exemplar.

Example: Given the exemplar vector e(1)=(-1 1 1 -1) and e(2)=(1 -1 1 -1). Use Hamming net to find the exemplar vector close to bipolar input patterns
(1 1 -1 -1), (1 -1 -1 -1), (-1 -1 -1 1) and (-1 -1 1 1).



Solution:

Step 1: Store the exemplars in the weights as:

wij = ei(j)/2 = i'th component of the j'th exemplar,

$$W = \begin{bmatrix} -0.5 & 0.5 \\ 0.5 & -0.5 \\ 0.5 & 0.5 \\ -0.5 & -0.5 \end{bmatrix}$$

Since e(1) = (-1 1 1 -1) and e(2) = (1 -1 1 -1).

bj = n/2 = 2

Step 2: Apply 1st bipolar input (1 1 -1 -1)

Yin1 = b1 + ∑ xi wi1

= 2 + (1 1 -1 -1) .* (-0.5 0.5 0.5 -0.5)

= 2

Yin2 = b2 + ∑ xi wi2

= 2 + (1 1 -1 -1) .* (0.5 -0.5 0.5 -0.5)

= 2

Hence, the first input patter has the same Hamming distance HD = 2 with both exemplar vectors.

Step 3: Apply the second input vector (1 -1 -1 -1)

Yin1 = 2 + (1 -1 -1 -1) .* (-0.5 0.5 0.5 -0.5) =1

Yin2 = 2 + (1 -1 -1 -1) .* (0.5 -0.5 0.5 -0.5) =3

Since y2 > y1, then the second input best matches with the second exemplar e(2).

Step 4: Apply input pattern no. 3 (-1 -1 -1 1)

Yin1 = 2 + (-1 -1 -1 1) .* (-0.5 0.5 0.5 -0.5) = 1

Yin2 = 2 + (-1 -1 -1 1) .* (0.5 -0.5 0.5 -0.5) = 1

Hence we have Hamming similarity.

Step 5: Consider the last input vector (-1 -1 1 1)

Yin1 = 2 + (-1 -1 1 1) .* 0.5 (-1 1 1 -1) = 2

Yin2 = 2+ (-1 -1 1 1) .* 0.5 (1 -1 1 -1) = 2

Hence we have Hamming similarity.