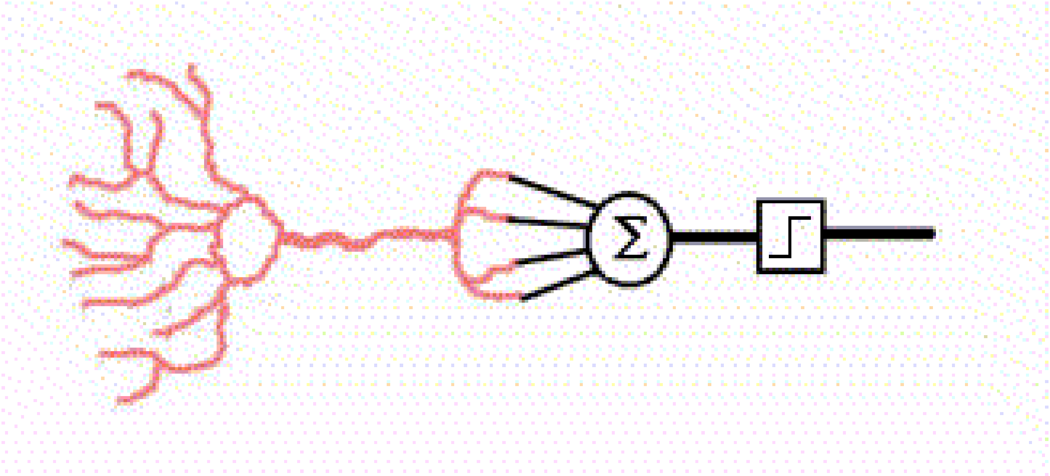


NEURAL NETWORKS (CS010 805G02)

Mod 4 -- Competitive networks

Shiney Thomas
AP,CSE,AJCE

Neural Networks Based on Competition

- Introduction
- Fixed weight competitive nets
  - Maxnet
  - Mexican Hat
  - Hamming Net
- Kohonen Self-Organizing Maps (SOM)
- Counterpropagation
- Adaptive Resonance Theory (ART)

COMPETITIVE NETS

Unsupervised

- MAXNET
- Hamming Net
- Mexican Hat Net
- Self-Organizing Map (SOM)
- Adaptive Resonance Theory (ART)

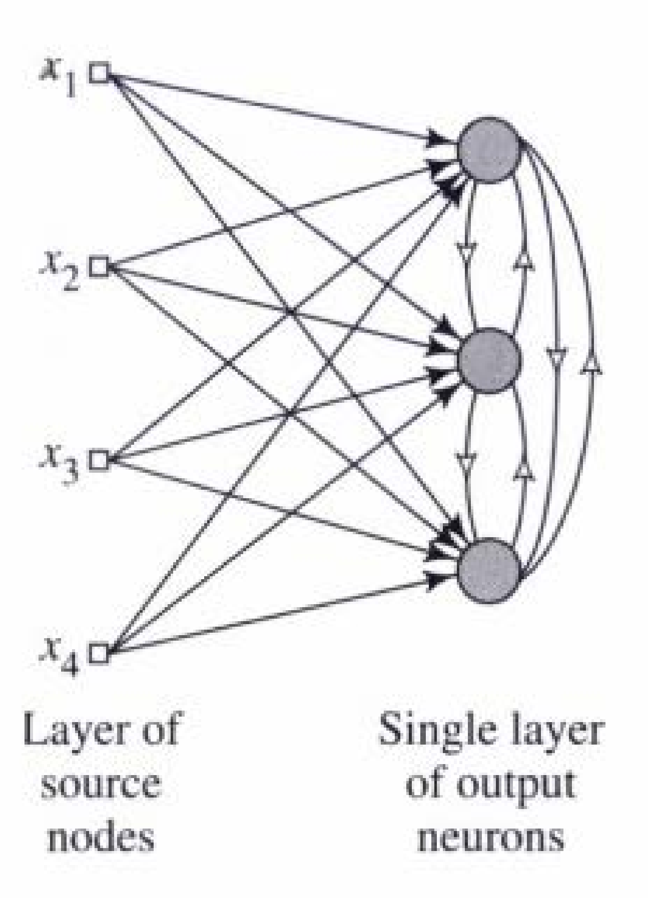
Supervised

- Learning Vector Quantization (LVQ)
- Counterpropagation

Competitive Learning

- Output neurons compete with each other for a chance to become active(fired).
- Highly suited to discover statistically salient features (that may aid in classification).
- Three basic elements:
  - Same type of neurons with different weight sets, so that they respond differently to a given set of inputs.
  - A limit imposed on the “strength” of each neuron.
  - Competition mechanism, to choose one winner: **winner-takes-all neuron** (WTA)

General Architecture of Comp. Learning Network



General Architecture of Comp. Learning Network

- In simplest form ,NN has a single layer of output neurons, each of which is fully connected to the input nodes
- feedforward connections are excitatory,
- and feedback connections perform lateral inhibition
- Winner selection

$$y_k = \begin{cases} 1 & \text{if } v_k > v_j \text{ for all } j, j \neq k \\ 0 & \text{otherwise} \end{cases}$$

Competitive Learning Network

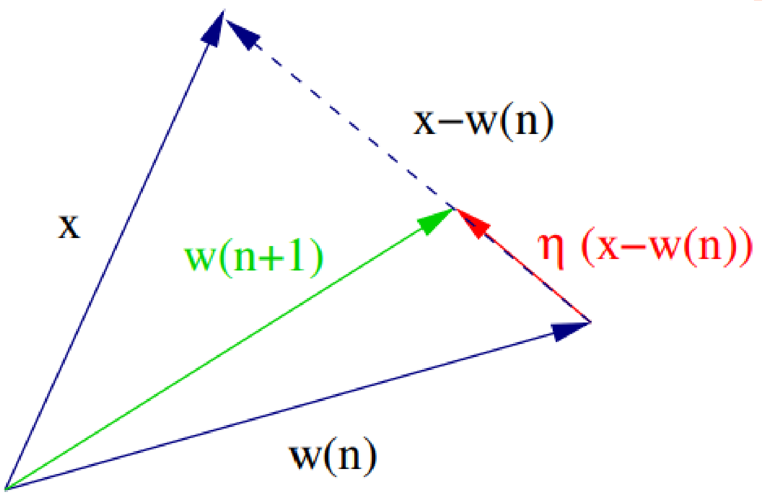
- Limit:

$$\sum_j w_{kj} = 1 \text{ for all } k.$$

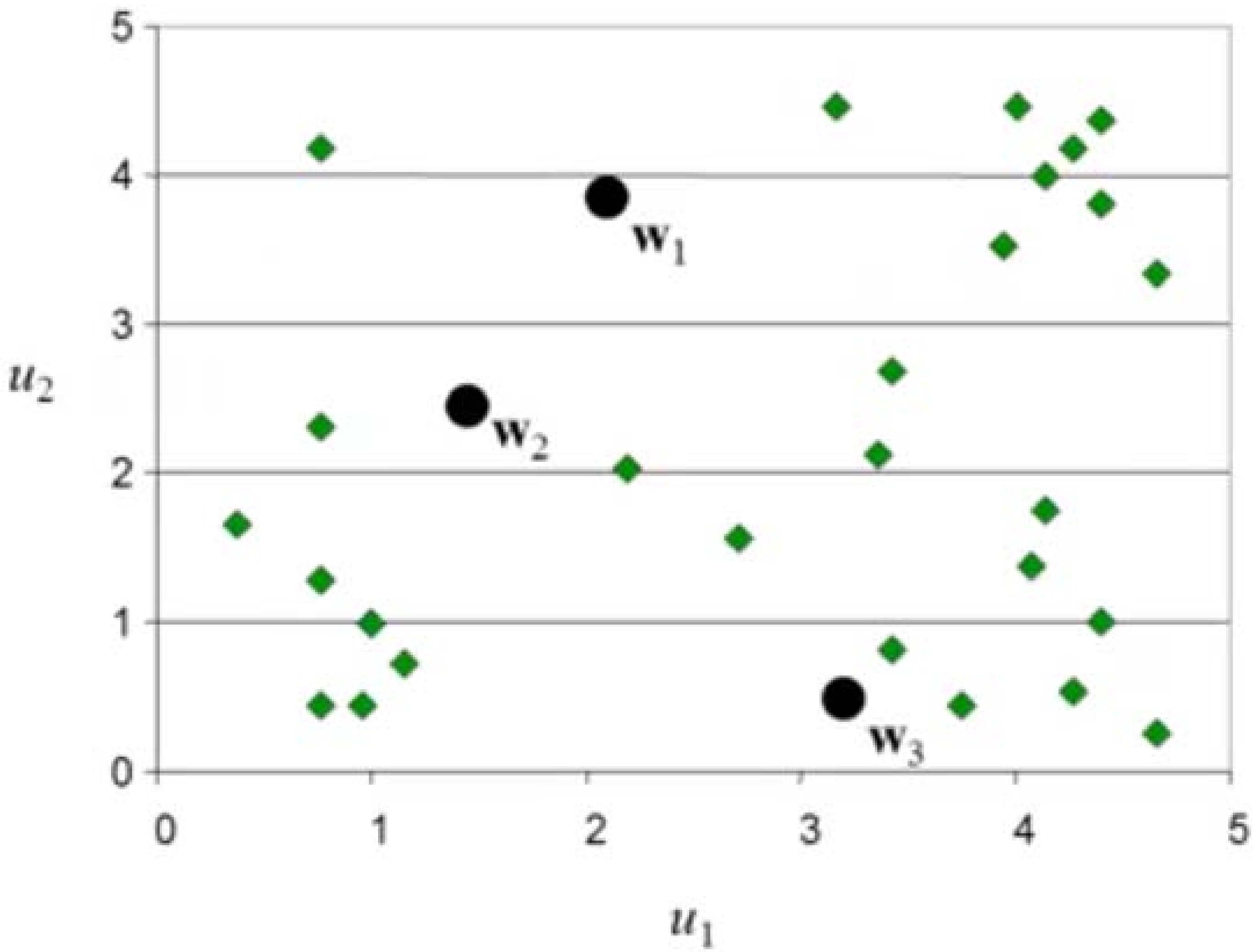
- Weight Change Adaptation:

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_k j) & \text{if } k \text{ is the winner} \\ 0 & \text{otherwise} \end{cases}$$

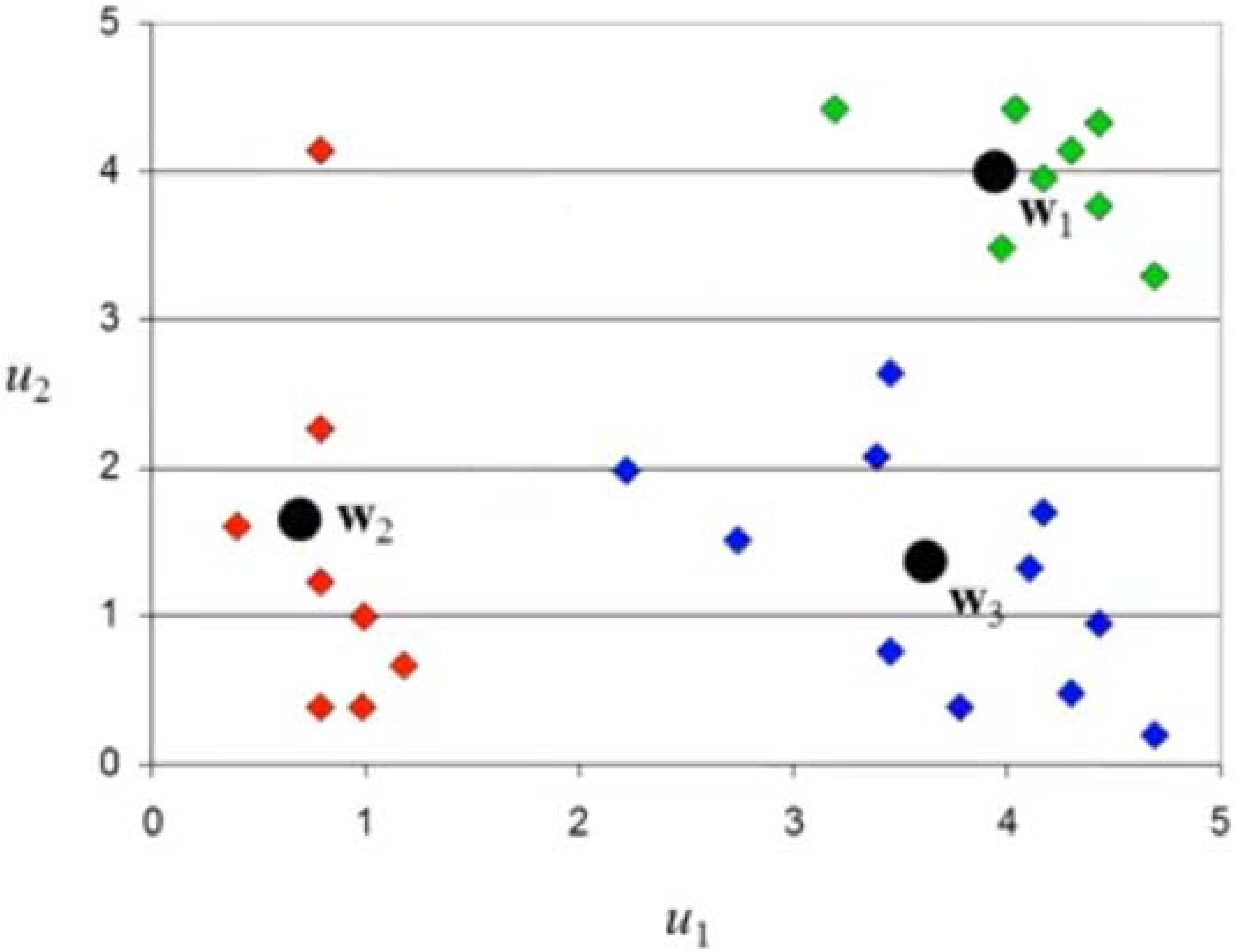
- The synaptic weight vector $w_k = (w_{k1}, w_{k2}, \ldots, w_{kn})$ is moved toward the input vector.



Competitive Learning Example-Initial Weights



Competitive Learning Example- After Updates



Competitive Nets

- The most extreme form of competition among a group of neurons is called "*Winner Take All*".
- As the name suggests, only one neuron in the competing group will have a nonzero output signal when the competition is completed.
- A specific competitive net that performs Winner Take All (WTA) competition is the "Maxnet".
- A more general form of competition, the "Mexican Hat" will also be described later.

Competitive Nets

- All of the other nets we will discuss use WTA competition as part of their operation.
- With the exception of the fixed-weight competitive nets (namely Maxnet, Mexican Hat, and Hamming net) all of the other nets combine <u>competition</u> with <u>some form of learning</u> to adjusts the weights of the net (i.e. *the weights that are not part of any interconnections in the competitive layer*).

COMPETITIVE NETS

- The form of learning depends on the purpose for which the net is being trained:
  - LVQ and counterpropagation net are trained to perform mappings. The learning in this case is
  - supervised.
  - SOM (used for clustering of input data): a common use of unsupervised learning.
  - ART are also clustering nets: also unsupervised.
- Several of the nets discussed use the same learning algorithm known as "*Kohonen learning*": where the units that update their weights do so by forming a new weight vector that is a linear combination of the old weight vector and the current input vector.
- Typically, the unit whose weight vector was closest to the input vector is allowed to learn.

Competitive Nets

- The weight update for output (or cluster) unit *j* is given as:

$$\mathbf{w}_{.j}(\text{new}) = \mathbf{w}_{.j}(\text{old}) + \alpha\left[\mathbf{x} - \mathbf{w}_{.j}(\text{old})\right]$$
$$= \alpha\mathbf{x} + (1-\alpha)\mathbf{w}_{.j}(\text{old})$$

where $\mathbf{x}$ is the input vector, $\mathbf{w}_{.j}$ is the weight vector for unit j, and α the learning rate, decreases as learning proceeds.

Competitive Nets

- Two methods of determining the closest weight vector to a pattern vector are commonly used for self-organizing nets.
- Both are based on the assumption that the weight vector for each cluster (output) unit serves as an exemplar for the input vectors that have been assigned to that unit during learning.
  - The first method of determining the winner uses the squared Euclidean distance between the I/P vector and the weight vector and chooses the unit whose weight vector has the smallest Euclidean distance from the I/P vector.
  - The second method uses the dot product of the I/P vector and the weight vector. The dot product can be interpreted as giving the correlation between the I/P and weight vector.

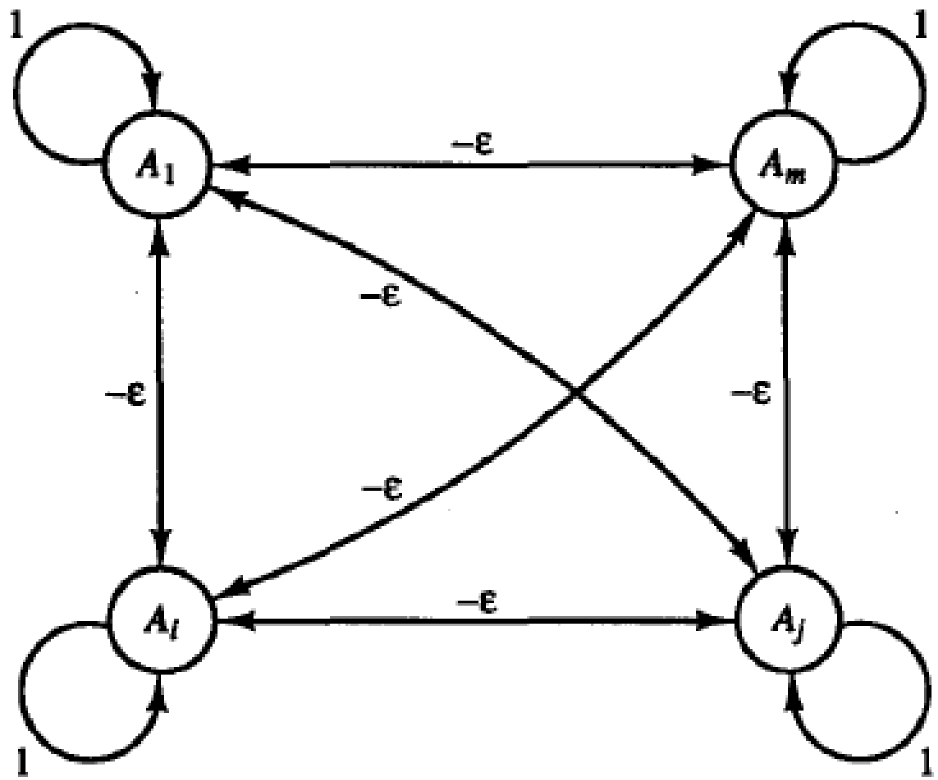
MAXNET

- Lippman, 1987
- Fixed-weight competitive net.
- Can be used as a subnet to pick the node whose input is largest
- m nodes in the subnet are completely interconnected, with symmetric weights
  - A recurrent network involving both excitatory and inhibitory connections
- Positive self-feedbacks and negative cross-feedbacks
- After a number of recurrences, the only non- zero node will be the one with the largest initializing entry from i/p vector

Maxnet Architecture

- Activation function for Maxnet is

$$f(x) = \begin{cases} x & \text{if } x \geq 0; \\ 0 & \text{otherwise.} \end{cases}$$



Maxnet Application

Step 0. Initialize activations and weights (set $0 < \epsilon < \frac{1}{m}$):

$a_j(0)$ input to node A_j,

$$w_{ij} = \begin{cases} 1 & \text{if } i = j; \\ -\epsilon & \text{if } i \neq j. \end{cases}$$

Step 1. While stopping condition is false, do Steps 2–4.

Step 2. Update the activation of each node: For $j = 1, \ldots, m$,

$$a_j(\text{new}) = f[a_j(\text{old}) - \epsilon \sum_{k \neq j} a_k(\text{old})].$$

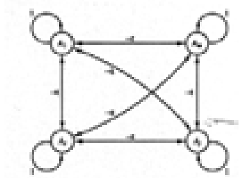
Step 3. Save activations for use in next iteration:

$$a_j(\text{old}) = a_j(\text{new}), j = 1, \ldots, m.$$

Step 4. Test stopping condition:
If more than one node has a nonzero activation, continue; otherwise, stop.

MAXNET EXAMPLE

- Note that in step 2, the i/p to the function *f* is the total i/p to node A_j from all nodes, including itself.
- Some precautions should be incorporated to handle the situation in which two or more units have the same, maximal, input.
- Example:



ε = .2 and the initial activations (input signals) are:

$a_1(0) = .2$, $a_2(0) = .4$ $a_3(0) = .4$ $a_4(0) = .8$

As the net iterates, the activations are:

The only node to remain on

$a_1(1) = f\{a_1(0) - .2\,[a_2(0) + a_3(0) + a_4(0)]\} = f(-.12) = 0$

$a_1(1) = .0$, $a_2(1) = .08$ $a_3(1) = .32$ $a_4(1) = .56$

$a_1(2) = .0$, $a_2(2) = .0$ $a_3(2) = .192$ $a_4(2) = .48$

$a_1(3) = .0$, $a_2(3) = .0$ $a_3(3) = .096$ $a_4(3) = .442$

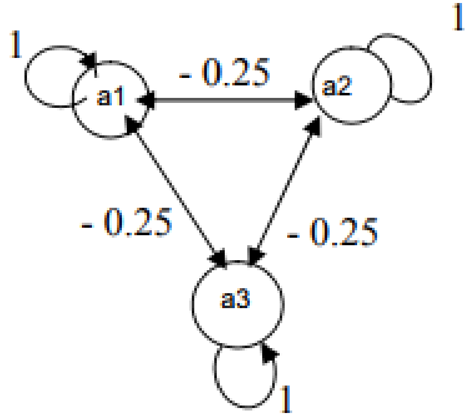
$a_1(4) = .0$, $a_2(4) = .0$ $a_3(4) = .008$ $a_4(4) = .422$

$a_1(5) = .0$, $a_2(5) = .0$ $a_3(5) = .0$ $a_4(5) = .421$

MAXNET EXAMPLE

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.

Maxnet example

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

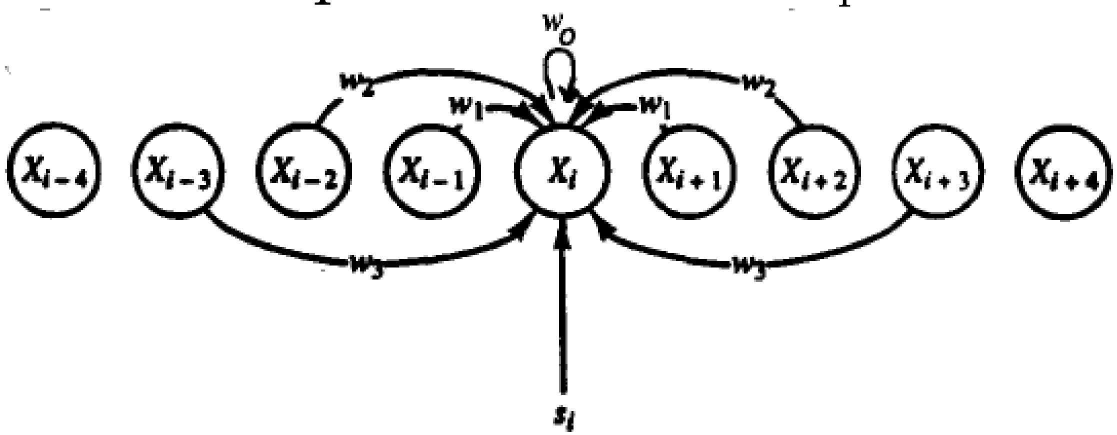
Then the 3rd neuron is the winner.

Mexican Hat

- Kohonen,1989
- A more general contrast enhancing subnet than Maxnet
- Each neuron is connected with excitatory links (positively weighted) to a number of "cooperative neighbors" neurons that are in close proximity.
- Each neuron is also connected with inhibitory links (with negative weights) to a number of "competitive neighbors" neurons that are somewhat further away.
- There may also be a number of neurons, further away still, to which the neurons is not connected.

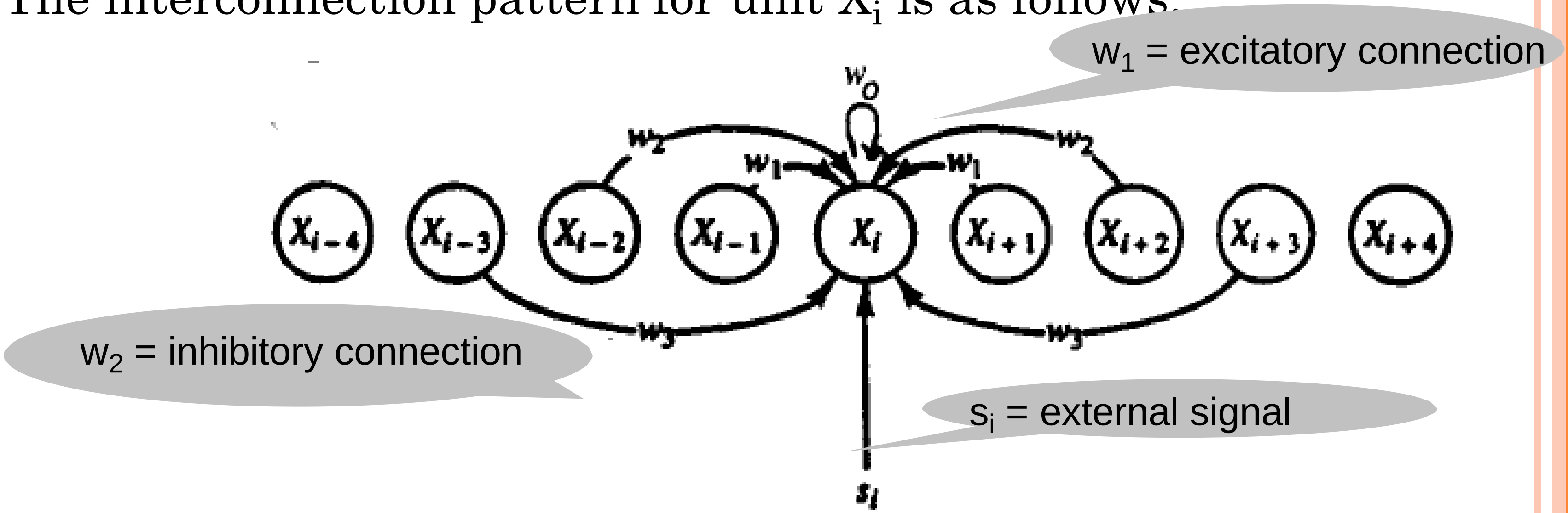
Mexican Hat

- All of these connections are within a particular layer of a neural net.
- The neurons receive an external signal in addition to these interconnections signals (just like Maxnet).
- This pattern of interconnections is repeated for each neuron in the layer.
- The interconnection pattern for unit X_i is as follows:



Mexican Hat Architecture

- The interconnection pattern for unit X_i is as follows:



- The contrast enhancement of the signal $\mathbf{s_i}$ received by unit Xi is accomplished by iteration for several time steps.
- The activation of unit $\mathbf{X_i}$ at time $\mathbf{t}$ is given by:

$$x_i(t) = f[s_i(t) + \sum_k w_k x_{i+k}(t-1)],$$

Mexican Hat Architecture

- The size of the region of cooperation (positive connections) and the region of competition (negative connections) may vary, as may vary the relative magnitudes of the +ve and –ve weights and the topology of the regions (linear, rectangular, hexagonal, etc..)

Algorithm

The algorithm given here is similar to that presented by Kohonen [1989a]. The nomenclature we use is as follows:

R_2	Radius of region of interconnections; X_i is connected to units X_{i+k} and X_{i-k} for $k = 1, \ldots, R_2$.
R_1	Radius of region with positive reinforcement; $R_1 < R_2$.
$\mathbf{w}_k$	Weight on interconnections between X_i and units X_{i+k} and X_{i-k}: w_k is positive for $0 \leq k \leq R_1$, w_k is negative for $R_1 < k \leq R_2$.
x	Vector of activations.
x_old	Vector of activations at previous time step.
t_max	Total number of iterations of contrast enhancement.
s	External signal.

Algorithm

Step 0. Initialize parameters t_max, R_1, R_2 as desired.
Initialize weights:

$$w_k = C_1 \text{ for } k = 0, \ldots, R_1 \ (C_1 > 0)$$

$$w_k = C_2 \text{ for } k = R_1 + 1, \ldots, R_2 \ (C_2 < 0).$$

Initialize **x_old** to **0**.

Step 1. Present external signal **s**:

$$\mathbf{x} = \mathbf{s}.$$

Save activations in array **x_old** (for $i = 1, \ldots, n$):

$$x\_old_i = x_i.$$

Set iteration counter: $t = 1$.

Step 2. While t is less than t_max, do Steps 3–7.

Step 3. Compute net input ($i = 1, \ldots, n$):

Algorithm

$$x_i = C_1 \sum_{k=-R_1}^{R_1} x\_old_{i+k}$$

$$+ C_2 \sum_{k=-R_2}^{-R_1-1} x\_old_{i+k} + C_2 \sum_{k=R_1+1}^{R_2} x\_old_{i+k}.$$

Step 4. Apply activation function (ramp function from 0 to x_max, slope 1):

$$x_i = \min(x\_max, \max(0, x_i)) \ (i = 1, \ldots, n).$$

Step 5. Save current activations in **x_old**:

$$x\_old_i = x_i \ (i = 1, \ldots, n).$$

Step 6. Increment iteration counter:

$$t = t + 1.$$

Step 7. Test stopping condition:
If $t < t\_max$, continue; otherwise, stop.

Application

Example 4.2 Using the Mexican Hat Algorithm

We illustrate the Mexican Hat algorithm for a simple net with seven units. The activation function for this net is

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

Step 0. Initialize parameters:

$$R_1 = 1;$$

$$R_2 = 2;$$

$$C_1 = 0.6;$$

$$C_2 = -0.4.$$

Step 1. *($t = 0$).*

The external signal is (0.0, 0.5, 0.8, 1.0, 0.8, 0.5, 0.0), so

$$\mathbf{x} = (0.0, 0.5, 0.8, 1.0, 0.8, 0.5, 0.0).$$

Application

Save in x_old:

x_old = (0.0, 0.5, 0.8, 1.0, 0.8, 0.5, 0.0).

Step 2. *(t = 1).*

The update formulas used in Step 3 are listed as follows for reference:

$$x_1 = 0.6\, x\_old_1 + 0.6\, x\_old_2 - 0.4\, x\_old_3$$

$$x_2 = 0.6\, x\_old_1 + 0.6\, x\_old_2 + 0.6\, x\_old_3 - 0.4\, x\_old_4$$

$$x_3 = -0.4\, x\_old_1 + 0.6\, x\_old_2 + 0.6\, x\_old_3 + 0.6\, x\_old_4 - 0.4\, x\_old_5$$

$$x_4 = -0.4\, x\_old_2 + 0.6\, x\_old_3 + 0.6\, x\_old_4 + 0.6\, x\_old_5 - 0.4\, x\_old_6$$

$$x_5 = -0.4\, x\_old_3 + 0.6\, x\_old_4 + 0.6\, x\_old_5 + 0.6\, x\_old_6 - 0.4\, x\_old_7$$

$$x_6 = -0.4\, x\_old_4 + 0.6\, x\_old_5 + 0.6\, x\_old_6 + 0.6\, x\_old_7$$

$$x_7 = -0.4\, x\_old_5 + 0.6\, x\_old_6 + 0.6\, x\_old_7.$$

APPLICATION

Step 3. $(t = 1)$.

$$x_1 = 0.6(0.0) + 0.6(0.5) - 0.4(0.8) = -0.2$$

$$x_2 = 0.6(0.0) + 0.6(0.5) + 0.6(0.8) - 0.4(1.0) = 0.38$$

$$x_3 = -0.4(0.0) + 0.6(0.5) + 0.6(0.8) + 0.6(1.0) - 0.4(0.8) = 1.06$$

$$x_4 = -0.4(0.5) + 0.6(0.8) + 0.6(1.0) + 0.6(0.8) - 0.4(0.5) = 1.16$$

$$x_5 = -0.4(0.8) + 0.6(1.0) + 0.6(0.8) + 0.6(0.5) - 0.4(0.0) = 1.06$$

$$x_6 = -0.4(1.0) + 0.6(0.8) + 0.6(0.5) + 0.6(0.0) = 0.38$$

$$x_7 = -0.4(0.8) + 0.6(0.5) + 0.6(0.0) = -0.2.$$

Step 4.

$$\mathbf{x} = (0.0,\ 0.38,\ 1.06,\ 1.16,\ 1.06,\ 0.38,\ 0.0).$$

Steps 5–7. Bookkeeping for next iteration.

Application

Step 3. $(t = 2)$.

$$x_1 = 0.6(0.0) + 0.6(0.38) - 0.4(1.06) = -0.196$$

$$x_2 = 0.6(0.0) + 0.6(0.38) + 0.6(1.06) - 0.4(1.16) = 0.39$$

$$x_3 = -0.4(0.0) + 0.6(0.38) + 0.6(1.06) + 0.6(1.16) - 0.4(1.06) = 1.14$$

$$x_4 = -0.4(0.38) + 0.6(1.06) + 0.6(1.16) + 0.6(1.06) - 0.4(0.38) = 1.66$$

$$x_5 = -0.4(1.06) + 0.6(1.16) + 0.6(1.06) + 0.6(0.38) - 0.4(0.0) = 1.14$$

$$x_6 = -0.4(1.16) + 0.6(1.06) + 0.6(0.38) + 0.6(0.0) = 0.39$$

$$x_7 = -0.4(1.06) + 0.6(0.38) + 0.6(0.0) = -0.196$$

Step 4.

$$\mathbf{x} = (0.0, 0.39, 1.14, 1.66, 1.14, 0.39, 0.0).$$

Steps 5–7. Bookkeeping for next iteration.

- Results for MH eg.