Unsupervised learning in ANN

Introduction

- **Unsupervised learning** is a type of self-organized Hebbian **learning** that helps find previously unknown patterns in data set without pre-existing labels.
- It is also known as self-organization and allows modeling probability densities of given inputs.
- **Unsupervised learning** is the training of an artificial **intelligence** (AI) algorithm using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance.
- **Unsupervised learning** algorithms can perform more complex processing tasks than supervised **learning** systems.

Fixed weight competitive nets

- In classification and prediction problems, we are provided with training sets with desired outputs, so backpropagation together with feed-forward networks are useful in modeling the input-output relationship.
- However, sometimes we have to analyze raw data of which we have no prior knowledge.
- The only possible way is to find out special features of the data and arrange the data in clusters so that elements that are similar to each other are grouped together.
- Such a process can be readily performed using simple competitive networks.
- Simple competitive networks are composed of two networks: the Hemming net and the Maxnet. Each of them specializes in a different function:

1.	The Hemming net measures how much the input vector resembles the weight vector of each perceptron.
2.	The maxnet finds the perceptron with the maximum value.

Winner-Takes-All Networks

- These kinds of networks are based on the competitive learning rule and will use the strategy where it chooses the neuron with the greatest total inputs as a winner.

- The connections between the output neurons show the competition between them and one of them would be 'ON' which means it would be the winner and others would be 'OFF'.

Hamming Network

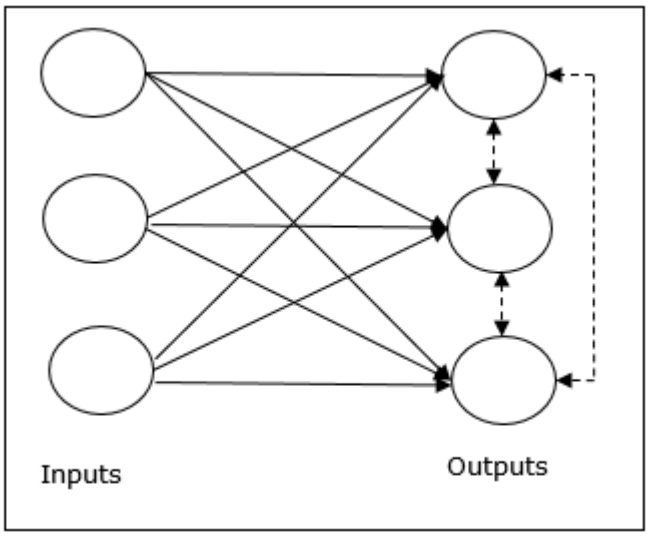
- In most of the neural networks using unsupervised learning, it is essential to compute the distance and perform comparisons.
- This kind of network is Hamming network, where for every given input vectors, it would be clustered into different groups.
- Following are some important features of Hamming Networks –
    1. Lippmann started working on Hamming networks in 1987.
    2. It is a single layer network.
    3. The inputs can be either binary {0, 1} of bipolar {-1, 1}.
    4. The weights of the net are calculated by the exemplar vectors.
    5. It is a fixed weight network which means the weights would remain the same even during training.

Competitive Learning in ANN

- It is concerned with unsupervised training in which the output nodes try to compete with each other to represent the input pattern.

Basic Concept of Competitive Network

- This network is just like a single layer feed-forward network having feedback connection between the outputs. The connections between the outputs are inhibitory type, which is shown by dotted lines, which means the competitors never support themselves.



Basic Concept of Competitive Learning Rule

- As said earlier, there would be competition among the output nodes so the main concept is - during training, the output unit that has the highest activation to a given input pattern, will be declared the winner.
- This rule is also called Winner-takes-all because only the winning neuron is updated and the rest of the neurons are left unchanged.

Mathematical Formulation

Following are the three important factors for mathematical formulation of this learning rule –

- **Condition to be a winner**

  Suppose if a neuron y_k wants to be the winner, then there would be the following condition

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

  It means that if any neuron, say, y_k wants to win, then its induced local field theoutputofthesummationunittheoutputofthesummationunit, say v_k, must be the largest among all the other neurons in the network.

- **Condition of the sum total of weight**

  Another constraint over the competitive learning rule is the sum total of weights to a particular output neuron is going to be 1. For example, if we consider neuron **k** then

$$\sum_k w_{kj} = 1 \quad for\ all\ k$$

- **Change of weight for the winner**

  If a neuron does not respond to the input pattern, then no learning takes place in that neuron. However, if a particular neuron wins, then the corresponding weights are adjusted as follows −

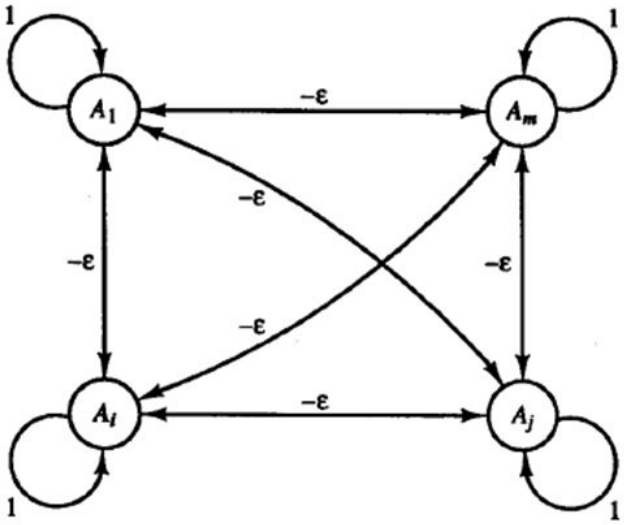
$$\Delta w_{kj} = \begin{cases} -\alpha(x_j - w_{kj}), & if\ neuron\ k\ wins \\ 0 & if\ neuron\ k\ losses \end{cases}$$

  Here αα is the learning rate.

Maxnet

- The maxnet is a fully connected network with each node connecting to every other nodes, including itself. The basic idea is that the nodes compete against each other by sending out inhibiting signals to each other.
- MAXNET is a specific example of a neural net based on competition.
- It can be used as a subnet to pick the node whose input is the largest.
- The m nodes in this subnet are completely interconnected, with symmetric weights.
- There is no training algorithm for the MAXNET; the weights are fixed.
- The architecture of MAXNET is as shown in the figure

MAXNET



The activation function for the MAXNET is

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

The application procedure is as follows:

1- Initialize activations and weights ($0 < \varepsilon < 1/m$)

$a_j(0)$ input to node $\mathbf{A}_j$

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

2- Update the activation of each node: j = 1, … , m

$$a_j(new) = f\,[a_j(old) - \varepsilon \sum_{k \neq j} a_k(old)]$$

3- Save activation for use in next iteration

$$a_j(old) = a_j(new), \quad j = 1, \ldots, m$$

4- If more than one node has a nonzero activation, then updating the activation will continue, otherwise we should stop.

Note that in Step 2, the input to the function *f* is simply the total input to node $\mathbf{A}_j$ from all nodes, including itself.

Example -1: Consider the action of a MAXNET with four neurons and inhibitory weights $\varepsilon = 0.2$. The neurons given initial activation (input signal)

$a_1(0) = 0.2, \quad a_2(0) = 0.4, \quad a_3(0) = 0.6, \quad a_4(0) = 0.8$

sol:

$$a_j(new) = f\,[a_j(old) - \varepsilon \sum a_k(old)]$$

$$a_1(new) = f\,[a_1(old) - 0.2(a_2(old) + a_3(old) + a_4(old)]$$

$$a_2(new) = f\,[a_2(old) - 0.2(a_1(old) + a_3(old) + a_4(old)]$$

$$a_3(new) = f\,[a_3(old) - 0.2(a_2(old) + a_2(old) + a_4(old)]$$

$$a_4(new) = f\,[a_4(old) - 0.2(a_1(old) + a_2(old) + a_3(old)]$$

$$a_1(1) = f\,[a_1(0) - 0.2(a_2(0) + a_3(0) + a_4(0)]$$

$$= f\,[0.2 - 0.2(0.4 + 0.6 + 0.8)] = f\,[-0.16] = 0$$

$$a_2(1) = f\,[a_2(0) - 0.2(a_1(0) + a_3(0) + a_4(0)]$$

$$= f\,[0.4 - 0.2(0.2 + 0.6 + 0.8)] = f\,[0.08] = 0.08$$

$$a_3(1) = f\,[a_3(0) - 0.2(a_1(0) + a_2(0) + a_4(0)]$$

$$= f\,[0.6 - 0.2(0.2 + 0.4 + 0.8)] = f\,[0.32] = 0.32$$

$$a_4(1) = f\,[a_4(0) - 0.2(a_1(0) + a_2(0) + a_3(0)]$$

$$= f\,[0.8 - 0.2(0.2 + 0.4 + 0.6)] = f\,[0.56] = 0.56$$

The activations found as the net iterates are

$a_1(1) = 0.0$	$a_2(1) = 0.08$	$a_3(1) = 0.32$	$a_4(1) = 0.56$
$a_1(2) = 0.0$	$a_2(2) = 0.0$	$a_3(2) = 0.192$	$a_4(2) = 0.48$
$a_1(3) = 0.0$	$a_2(3) = 0.0$	$a_3(3) = 0.096$	$a_4(3) = 0.442$
$a_1(4) = 0.0$	$a_2(4) = 0.0$	$a_3(4) = 0.008$	$a_4(4) = 0.422$
$a_1(5) = 0.0$	$a_2(5) = 0.0$	$a_3(5) = 0.0$	$a_4(5) = 0.421$