Unit-2

Unsupervised learning Network

In unsupervised learning, the machine is trained on a set of unlabeld data, which means that the input data is not paired with the disred output.

→ It is wed for tasks such as clustring, dimentionality Reduction etc.

→ It is to discover patterns and relationships in the data without any explicit guidance

Type of unsupervised learning Network

It is classified 2 categories

① clustring

A clustering is used to group the data. used for groupins the similar data points tougether.

clustering types

① Hierarchical clustring
② k-mean clustring
③ principal component clustring
④ Independent component Analysis

② Dimensionality Reduction

It reduces the number of features in a dataset while preserving its essential structure

Principal component Analysis

Transforms the data into a lower dimensional space by identifying the directions that maximize variance.

3. Association Rule learning

This involves discovering interesting relationships between variables in large datasets.

Applications

→ customer segmentation
→ Image compression.
→ Recommendation system
→ Natural language processing

Fixed Weight Competitive Nets

* It is neural network architecture that fall under the category of unsupervised learning.

* designed to model competitive learning, where neurons compete to respond to a given input.

* unlike Traditional neural networks weights it maintain, fixed weights.

Types

1. MAXNET
2. Hamming Network.

(1) MaxNet

It is a simple competitive neural networks where neurons compete to determine which one has the highest activation.

→ The neuron with the highest initial activation suppresses the activation of all other neurons, eventually only winning Neuron only remains.

Features

(1) fixed weights.

(2) winner - takes - All

(3) Recurrent connections
↳ iterated to suppress other neurons.

How it works

→ Each neuron receives input and computes its activation.

→ Neurons are connected to each other with (negative weights)

→ The neuron with the highest activation suppresses the activations of other neurons through these inhibitory connections.

→ over time, only the neuron with the highest initial activation remains active.

Ex:-

inputs [0.8, 0.6, 0.9, 0.7]

Neurons

Neuron 1 → (0.8)

" 2 → (0.6)

" 3 → (0.9)

" 4 → (0.7)

It checks the activation values and only with highest will survive like it keeps the '0' which drops and '1' which wins

[0, 0, 1, 0] final output.

(0.9) → survives.

Applications

→ selecting the best option.

→ Searching for the similar image or text from database.

2. Hamming Network

it is a two-layer neural network for pattern classification. it computes the Hamming distance between the input pattern and stored prototype patterns, and output layer determine the closest match.

Features

Two layers

1 Layer → computs the distance between the input and stored patterns

2 Layer → A maxNet that selects the pattern with the smallest Hamming distance.

Fixed weights → The weights in the first layer are fixed and represent the stored prototype patterns

Competitive learning → The Maxnet in the second layer ensures that only the closest pattern is selected.

How it works

- Layer 1 → The input pattern is compared to each stored prototype pattern, and the Hamming distance is computed.

Layer 2 → The maxnet takes the Hamming distances as inputs and selects the pattern

with the smallest distance is the winner.

Ex:-

input pattern : [1, 0, 0, 0]

Layer 1 → Hamming Distance calculation

Prototype patterns.

- Pattern 1 : [1, 0, 1, 0] → Hamming Distance = 1
- pattern 2 : [0, 1, 0, 1] → Hamming Distance = 3
- pattern 3 : [1, 1, 0, 0] → Hamming Distance = 1

Layer 2 → MaxNet competition

Input to maxnet [1, 3, 1]

MaxNet output : [1, 0, 1]

Pattern 1 & 3 are have smallest Hamm distance they have closest match

Hammins distance = compare both input and pattern so based on that value

Smallest Hamming distance is 1

[1, 0, 1]

Hammins distance Nearest Smallest activation choose chasulcorali.

Applications

→ Pattern recognition

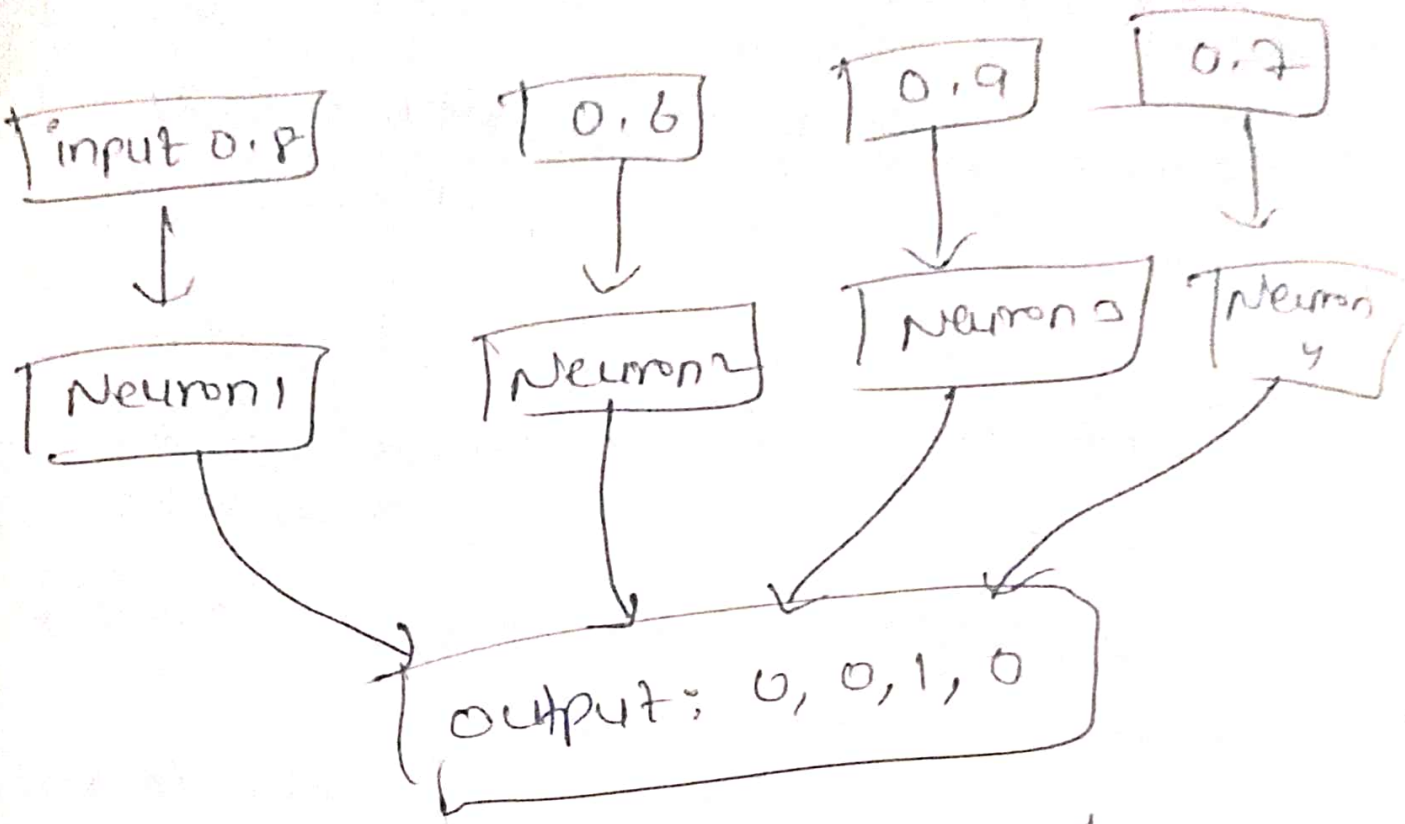
→ binary image or signals classification.

Comparision

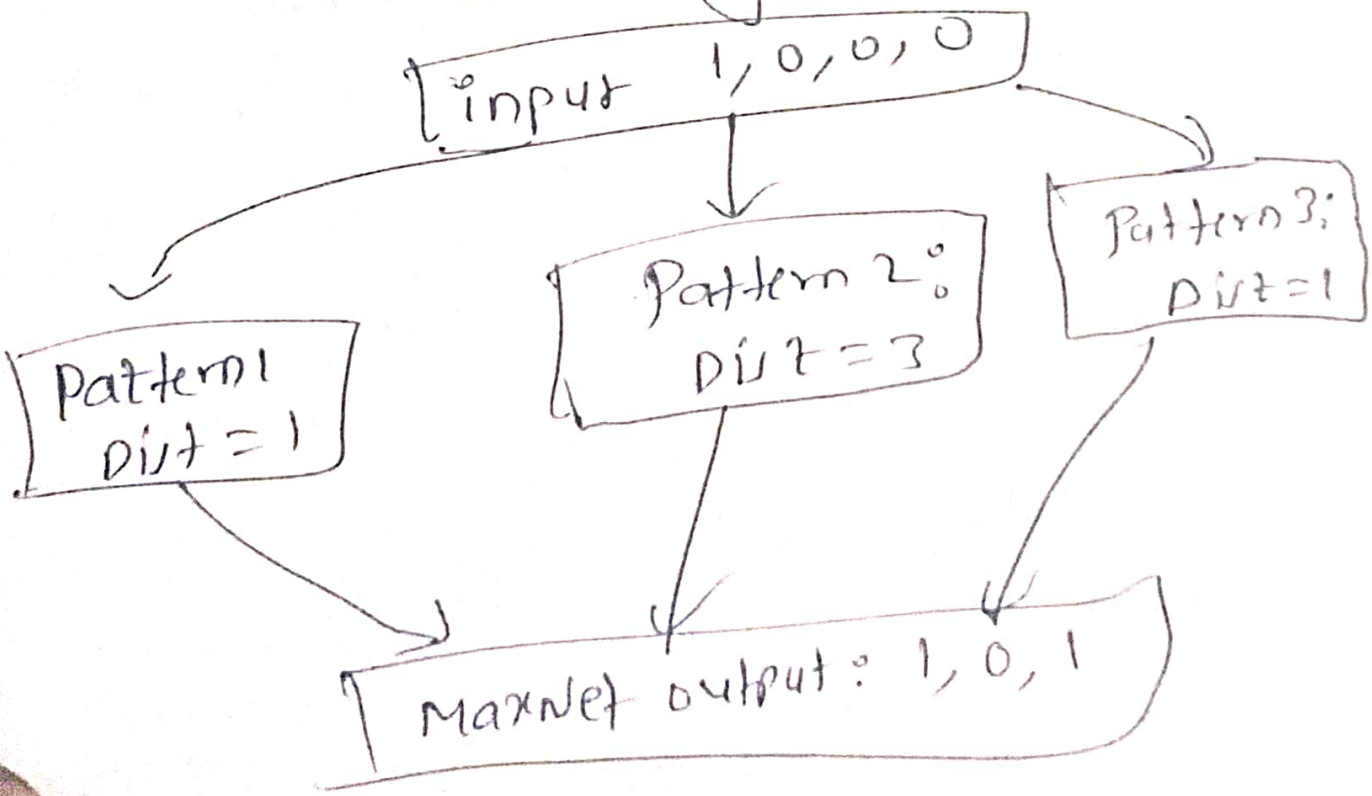
MaxNet	Hamming Network
select the highest activation	classifies input patterns.
single layer	two layers (Hamming + MaxNet)
Fixed inhibiton weights	Fixed weights
winning neuron	closest stored pattern.
slecting the best option	Pattern recognition.

Ex Diagrams of Previous Example

MAXNET



Hamming Network



Kohonen Self-organizing feature Maps

It is also known as Kohonen Networks, are a type of artificial neural networks that use unsupervised learning to produce a low dimensional representation of input data.

key things

(1) Unsupervised learning
↳ SOMs do not require labeled data.

(2) Topological preservation
↳ SOM's maintain the spatial relationships between data points, meaning similar inputs are mapped close to each other in the output space.

(3) competitive learning
Neurons in the SOM compete to be activated by input data, and the winning neuron updates its weights to become more similar to the input.

Structure

input layer

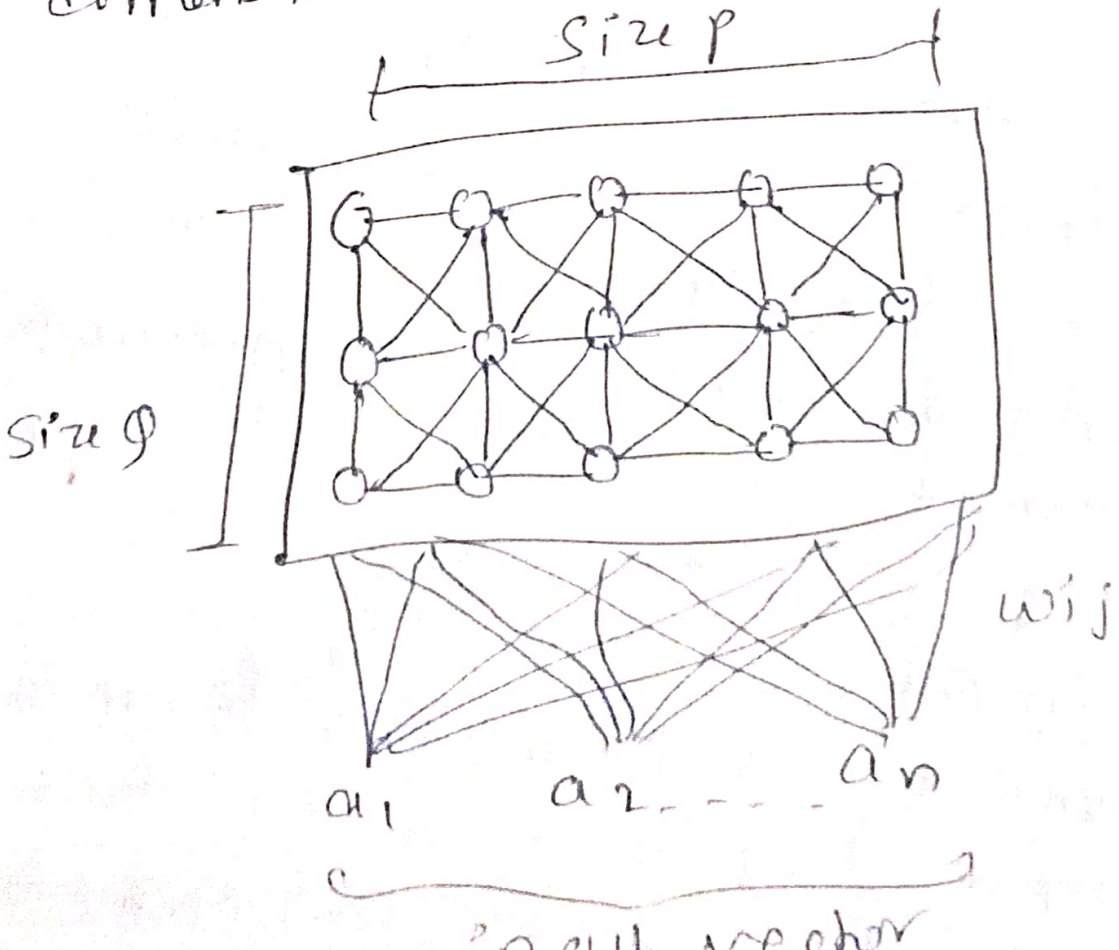
- Receives high-dimensional input data.

output layer (map)

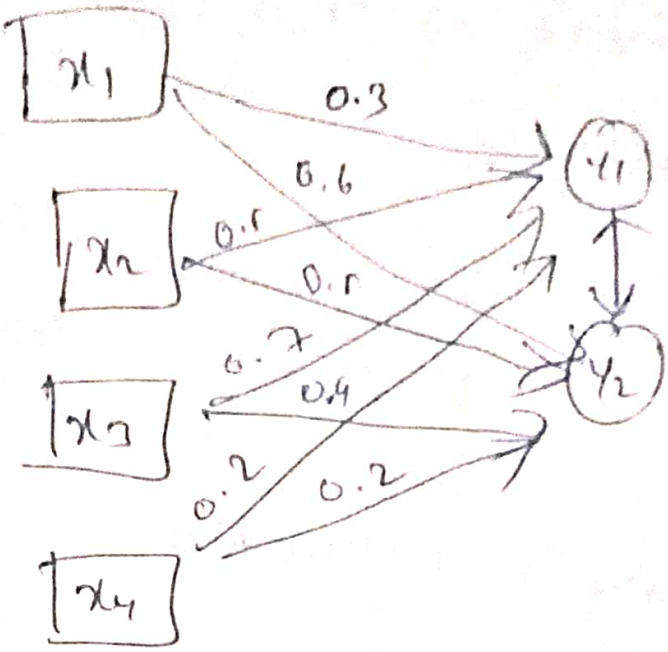
- A 2D grid of neurons, each representing a prototype vector

weights

- each neuron in the output layer has a weight vector of the same dimension as the input data.



Ex:-



Consider 4 training samples each vector of length 4 and two output units.

Training Samples

x_1: (1, 0, 1, 0)

x_2: (1, 0, 0, 0)

x_3: (1, 1, 1, 1)

x_4: (0, 1, 1, 0)

Ouput unit :- unit 1, unit 2

learning rate $\eta(t) = 0.6$

initial weight matrix

$$\begin{bmatrix} \text{unit 1} \\ \text{unit 2} \end{bmatrix} = \begin{bmatrix} 0.3 & 0.5 & 0.7 & 0.2 \\ 0.6 & 0.5 & 0.4 & 0.2 \end{bmatrix}$$

Iteration 1

Training sample x_1: (1, 0, 1, 0)

weight matrix

$$\begin{bmatrix} \text{unit 1} \\ \text{unit 2} \end{bmatrix} : \begin{bmatrix} 0.3 & 0.5 & 0.7 & 0.2 \\ 0.6 & 0.7 & 0.4 & 0.3 \end{bmatrix}$$

distance formula Euclidean

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

x_1: (1, 0, 1, 0) and unit 1 weights

$$d^2 = (0.3-1)^2 + (0.5-0)^2 + (0.7-1)^2 + (0.2-0)^2$$

$$= 0.87$$

x_1: (1, 0, 1, 0) and unit 2 weights

$$d^2 = (0.6-1)^2 + (0.7-0)^2 + (0.4-1)^2 + (0.3-0)^2$$

$$= 1.1$$

Here unit 1 wins ∵ it is smaller

Now we need to update weights

$$w_j(t+1) = w_j(t) + \eta(t)(x_i - w_j(t))$$

new weight — old weight — learning rate — input — old weight

unit 1 weights

$$= [0.3\ 0.1\ 0.7\ 0.2] + 0.6([1\ 0\ 1\ 0] - [0.3\ 0.1\ 0.7\ 0.2])$$

$$= [0.3\ 0.1\ 0.7\ 0.2] + 0.6[0.7\ -0.1\ 0.3\ -0.2]$$

$$= [0.3\ 0.1\ 0.7\ 0.2] + [0.42\ -0.30\ 0.18\ -0.12]$$

$$= [0.72\ 0.2\ 0.88\ 0.08]$$

$$\begin{bmatrix} \text{unit 1} \\ \text{unit 2} \end{bmatrix} = \begin{bmatrix} 0.72 & 0.2 & 0.88 & 0.08 \\ 0.6 & 0.7 & 0.4 & 0.3 \end{bmatrix}$$

Iteration 2

$x_2 : (1, 0, 0, 0)$

weight $\begin{bmatrix} \text{unit 1} \\ \text{unit 2} \end{bmatrix} : \begin{bmatrix} 0.72 & 0.2 & 0.88 & 0.08 \\ 0.6 & 0.7 & 0.4 & 0.3 \end{bmatrix}$

$x_2 : (1,0,0,0)$ and unit 1 weights

$d^2 = (0.72-1)^2 + (0.2-0)^2 + (0.88-0)^2$
$+ (0.08-0)^2 = 0.74$

$x_2 : (1,0,0,0)$ and unit 2 weights

$d^2 = (0.6-1)^2 + (0.7-0)^2 + (0.4-0)^2$
$+ (0.3-0)^2$

$= 0.9$

Unit 1 wins

update weights

new weights

$$\begin{bmatrix} \text{unit } 1 \\ \text{unit } 2 \end{bmatrix} : \begin{bmatrix} 0.89 & 0.08 & 0.37 & 0.03 \\ 0.6 & 0.7 & 0.4 & 0.3 \end{bmatrix}$$

Iteration 3

$x_1 : (1,1,1,1)$

$x_3 : (1,1,1,1)$ and unit 1

$d^2 = 2.2$

$x_3 : (1,1,1,1)$ and unit 2

$d^2 = 1.1$

Here unit 2 wins

update weights

$$\begin{bmatrix} \text{unit1} \\ \text{unit2} \end{bmatrix} = \begin{bmatrix} 0.89 & 0.01 & 0.37 & 0.03 \\ 0.84 & 0.11 & 0.76 & 0.72 \end{bmatrix}$$

Iteration 4

$x_4 : (0, 1, 1, 0)$

x_4: (0,1,1,0) and unit1

$d^2 = 2.06$

x_4: (0,1,1,0) and unit2

$d^2 = 1.3$

unit2 wins update weight

$$\begin{bmatrix} \text{unit1} \\ \text{unit2} \end{bmatrix} : \begin{bmatrix} 0.89 & 0.01 & 0.37 & 0.03 \\ 0.34 & 0.91 & 0.9 & 0.29 \end{bmatrix}$$

Best Mapping units for each sample

$x_1 : (1, 0, 1, 0) \rightarrow$ unit 1

$x_2 : (1, 0, 0, 0) \rightarrow$ unit 1

$x_3 : (1, 1, 1, 1) \rightarrow$ unit 2

$x_4 : (0, 1, 1, 0) \rightarrow$ unit 2

} epoch

Proceed is continued for many epoches until the feature map does not change.

Learning vector Quantization

* it is a supervised machine learning algorithm used for classification tasks. it is a type of artificial neural networks that builds on the concepts of self-organizing maps but operates in a supervised manner.

* The same researcher who developed SOM is introduced LVQ.

concepts

(1) Supervised learning
LVQ requires labeled data for training.

(2) prototype vectors
LVQ uses a set of prototype vectors, each representing a class.

(3) Competitive learning

↳ similar to SOMs, they use a competitive process to determine the closest prototype to an input vector.

(4) update Rule

↳ prototype are updated based on whether they correctly classify the input data.

Mathematical update Rule

if the prototype w_i correctly classifies the input x:

$$w_i(t+1) = w_i(t) + \alpha(t) \cdot (x - w_i(t))$$

if the prototype w_i incorrectly classifies the input x

$$w_i(t+1) = w_i(t) - \alpha(t) \cdot (x - w_i(t))$$

$w_i(t)$ is the prototype vector at time t.

$\alpha(t)$ is the learning rate

x is the input vector.

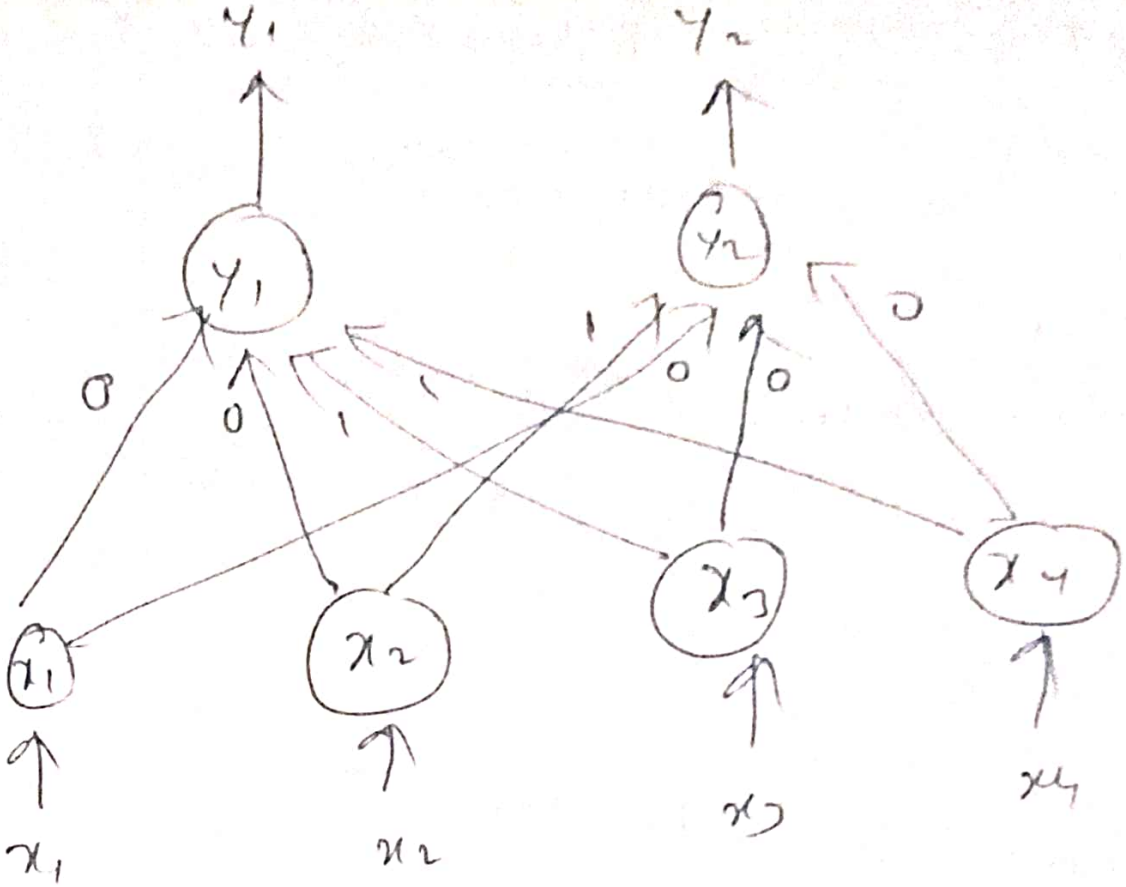
Construct and test an LVQ net with five vectors assigned to two classes.

The vector along with the classes are as

vector	class
[0,0,1,1]	1
[1,0,0,0]	2
[0,0,0,1]	2
[1,1,0,0]	1
[0,1,1,0]	1

Learning rate $\alpha = 0.1$

first we consider first two vectors as initial weights. and remaining consider as input vectors.



$w_1 = [0, 0, 1, 1]$

$w_2 = [1, 0, 0, 0]$

$\alpha = 0.1$

First input vector $[0, 0, 0, 1]$

with $\tau = 2$ (tarsit)

Calculate Euclidean distance

$$D(j) = \sum_{i=1}^{4} (w_{ij} - x_i)^2$$

$$D(j) = (w_{1j} - x_1)^2 + (w_{2j} - x_2)^2 + (w_{3j} - x_3)^2 + (w_{4j} - x_4)^2$$

$$D(1) = (w_{11} - x_1)^2 + (w_{21} - x_2)^2 + (w_{31} - x_3)^2 + (w_{41} - x_4)^2$$

$$D(1) = 1$$

Same way for $D_2 = 2$

$D(1) < D(2)$, Hence $j = 1$

Now $i \neq j$, the weights are updated. negative

$$w_{ij}(\text{New}) = w_{ij}(\text{old}) - \alpha(x_i - w_{ij}(\text{old}))$$

$$w_{11} = 0$$

$$w_{21} = 0$$

$$w_{31} = 1.1$$

$$w_{41} = 1$$

$$w_1 = [0, 0, 1.1, 1]$$

Second input vector [1, 1, 0, 0] with $\tau = 1$

$$D(j) = \sum_{i=1}^{4} (w_{ij} - x_i)^2$$

$D(1) = 4.21$

$D(2) = 1$

$D(2) < D(1)$ Hence $j = 2$

Now $\tau \neq j$ update weights

$w_{ij}(\text{New}) = w_{ij}(\text{old}) - \alpha(x_i - w_{ij}(\text{old}))$

$w_{12} = 1$

$w_{22} = 1 - 0.1$

$w_{32} = 0$

$w_{42} = 0$

$w_1 = [0, 0, 1, 1]$

$w_2 = [1, -0.1, 0, 0]$

Third input vector [0, 1, 1, 0]

with $\tau = 1$

Calculate euclidean distance

$D(1) = 2.01$

$D(2) = 3.21$

$D(1) < D(2)$, Hence $j = 1$

Now $i = j$, update the weights

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

$w_{11} = 0$

$w_{21} = 0.1$

$w_{31} = 1.09$

$w_{41} = 0.9$

$w_1 = [0, 0.1, 1.09, 0.9]$

$w_2 = [1, -0.1, 0, 0]$

after completing the first epoch. we should repeat this epoch until correct class is obtained for all input patterns

Counter Propagation Network

Counter Propagation Network are a type of ANN that combine supervised and unsupervised learning.

CPN are designed to perform tasks such as pattern recognition, data compression, function approximation.

Two layer architecture

Kohonen layer → Performs unsupervised learning to cluster input data.

Grossberg layer → Performs supervised learning to map the clustered data to the desired output.

→ This are computationally efficient and can handle large datasets.

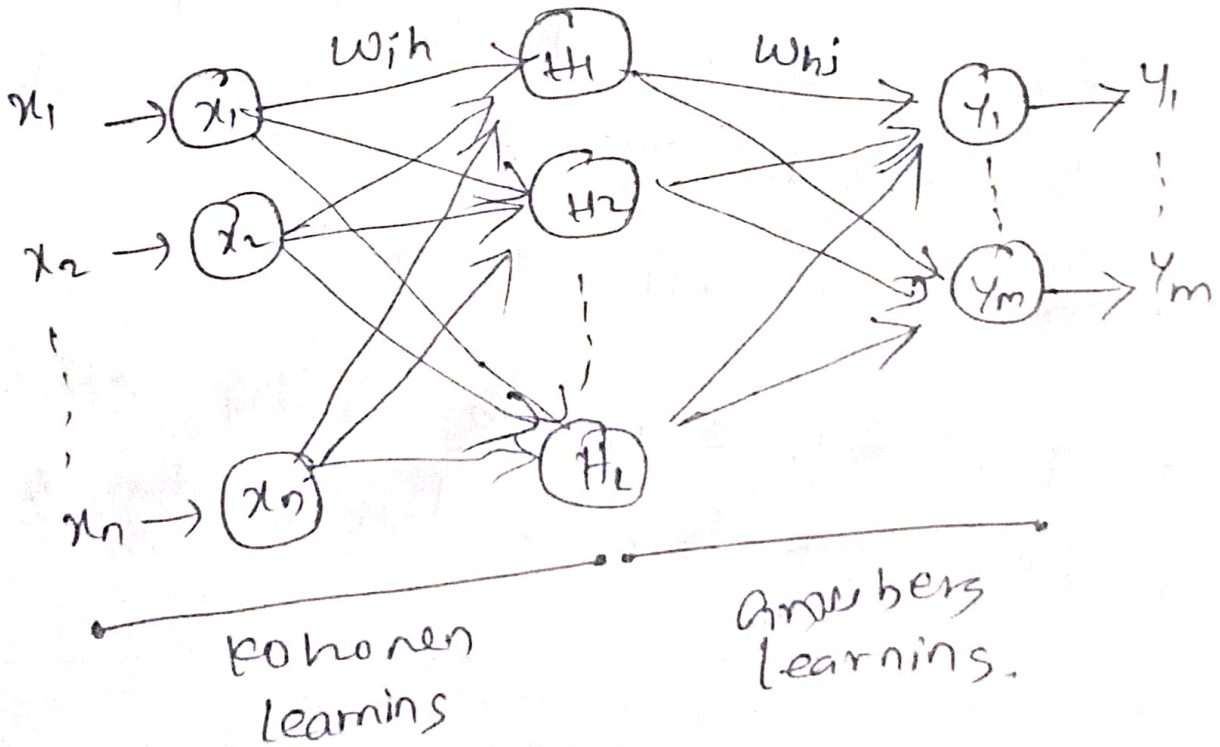
Two types

① full counter propagation Network (full cpn)

it is bidirectional network that can perform both forward mappings and (inp to out)
reverse mapping (out to inp)

Architecture

→ input layer
→ kohonen layer
→ Grossbergs layer.



Code

forward mapping

Inputs are passed through the kohonen layer to the Grossberg layer, which produces output.

Reverse mapping

Outputs are fed back into the Grossberg layer, which reconstructs the original input.

Applications

→ Data compression
→ Associative memory
→ Bidirectional mapping.

Limitations

→ complex
→ Requires more computational resources.

(2) Forward-only counterpropagation Network (forward-only CPN).

It is unidirectional network that performs only forward mappings it does not support reverse mapping.

Architecture

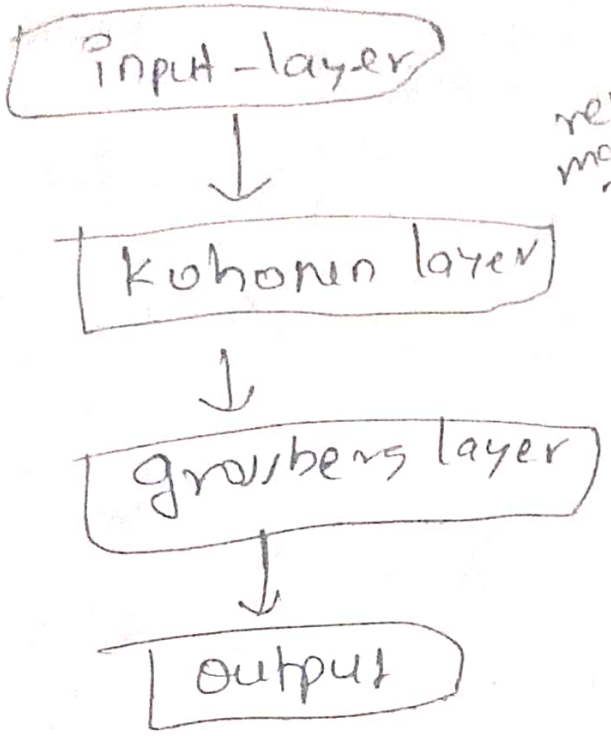
- input layer
- Kohonen layer
- Grossberg layer

working

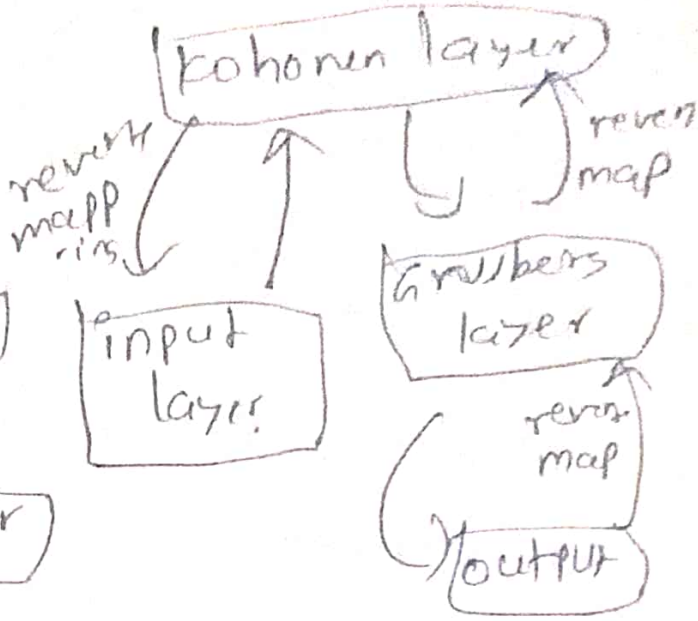
→ Inputs are passed through the Kohonen layer to the Grossberg layer, which produce the output.

→ No reverse mapping is performed.

forward-only CPN



Full CPN



Applications for forward-only

→ Pattern recognition
→ Function Approximation.
→ Supervised learning tasks forward is required.

Limitations

→ cannot perform reverse map
→ Limited to tasks where only input-to-output mapping needed.

Adaptive Resonance Theory Networks

Adaptive Resonance Theory is a family of NN developed by and used to solve the stability - plasticity dilemma. refer to like create a system

→ Adapt to new information (plasticity)

→ Retain previously learned knowledge (stability)

ART networks are unsupervised learning models used for clustering and pattern recognition.

Concepts

① Stability - Plasticity Balance

→ ART network can learn new patterns without overwriting or disrupting existing knowledge

→ This makes them highly efficient for incremental learning.

2. Vigilance Parameter

→ Key feature of ART Network is this (ρ), which controls the granularity of clustring.

→ a high vigilance parameter creates fine-grained clusters. low then coarse-grained clusters.

3. Resonance

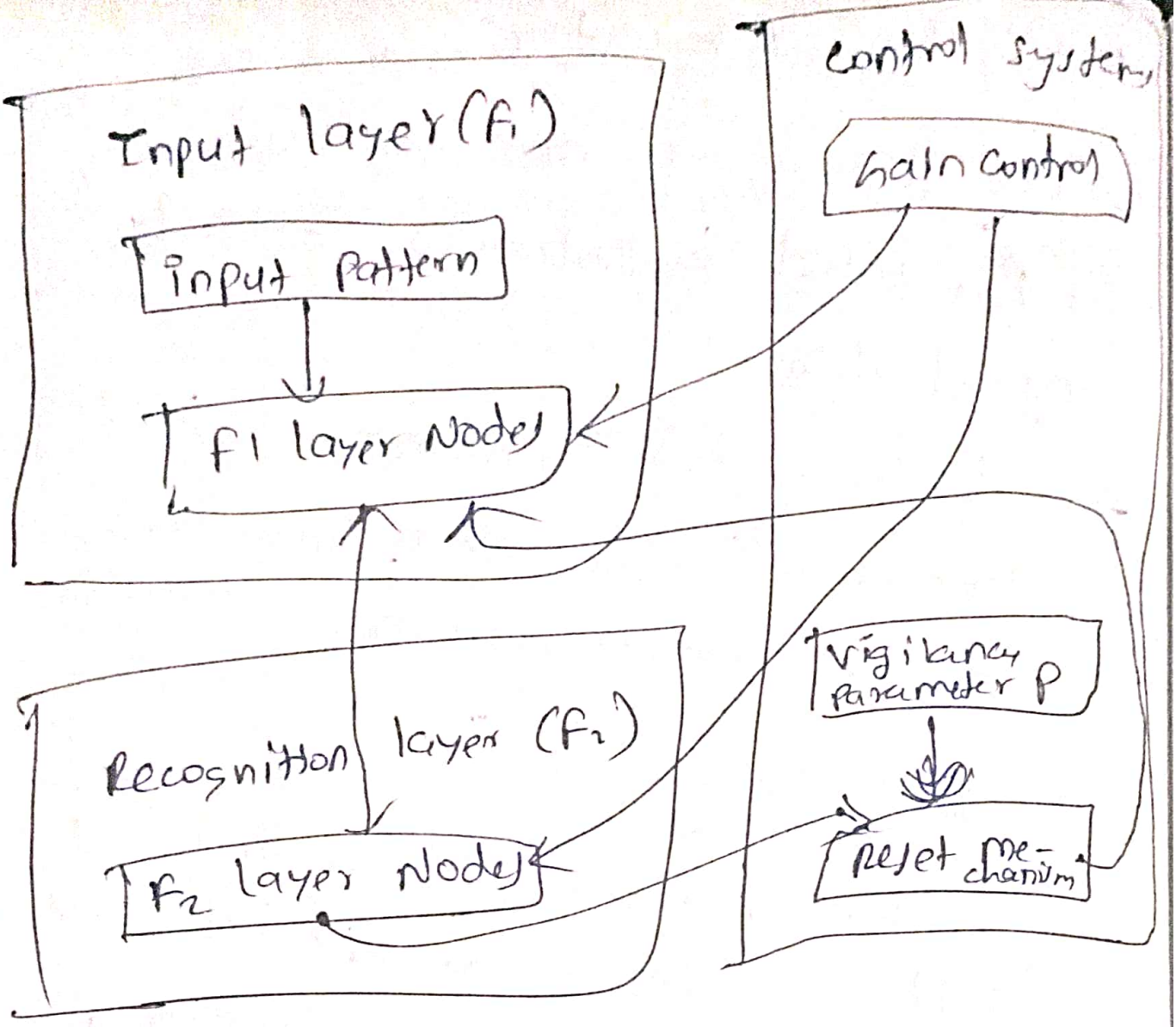
→ when an input pattern matches a stored category, the network enters a state of resonance, meaning the input is recognized and category is updated.

→ if no match, a new category is created.

4. Two layer Architecture

Input layer (F_1) → Receives and processes input patterns

output layer (F_2) → Represents the recognition or categories or clusters.



Types

① ART 1 → Designed for binary input data.

② ART 2 → Handles continuous-valued input data.

③ ART 3 → Extends ART 2 with additional biological plausibility.

④ Fuzzy ART → incorporates fuzzy logic to handle ambiguous or uncertain data.

Working

(1) Initialization

(2) input presentation

(3) category selection

(4) vigilance test

(5) Learning.

Ex:-

dataset of animals ↗ want to classified how the

Cat, Dogs, Birds.

input data

↳ feature vector Representation

cat [0.6, 0.2, 0.1]

initialization

Set the vigilance parameter $\rho = 0.7$

No category exist initially

First input (cat image) [0.6, 0.2, 0.1]

↳ No data exist, so create a new

category "Cat"

update the weights to represent

the cat.

Second input (Dog image)

input [0.1, 0.9, 0.7]

compare cat category similarity is low (β) create new category Dog.

Third image (Cat image)

input [0.7, 0.3, 0.6]

compare with cat category

Similarity is high (above β), so assign the cats and update weights.

Result

The ART network clustered the images in two categories cats, Dogs.

Advantages

① Incremental learning
② Stability - plasticity Balance
③ Real - time operation.
④ Customizable Granularity.

Application

1. Pattern Recognition
2. image and speech Recognition
3. Data mining
4. Robotics.

~~Simple~~

Special Networks - introduction to various Networks

8 common types of Neural Networks

1. Convolutional Neural Networks
2. Reurent Neural Networks
3. Long - short term memory Networks
4. Multilayer Perceptrons
5. Self-organizing maps.
6. Gated Recurrent Unit.
7. Generative Adversal Networks.
8. Transformers.

Same check all other chapters