

NPTEL ONLINE CERTIFICATION COURSES

Course Name: Fuzzy logic and Neural Networks

Faculty Name: Prof. D.K. Pratihar

Department: Mechanical Engineering

Week 6

NPTEL



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Fuzzy logic and Neural Networks

Faculty Name: Prof. D.K. Pratihar

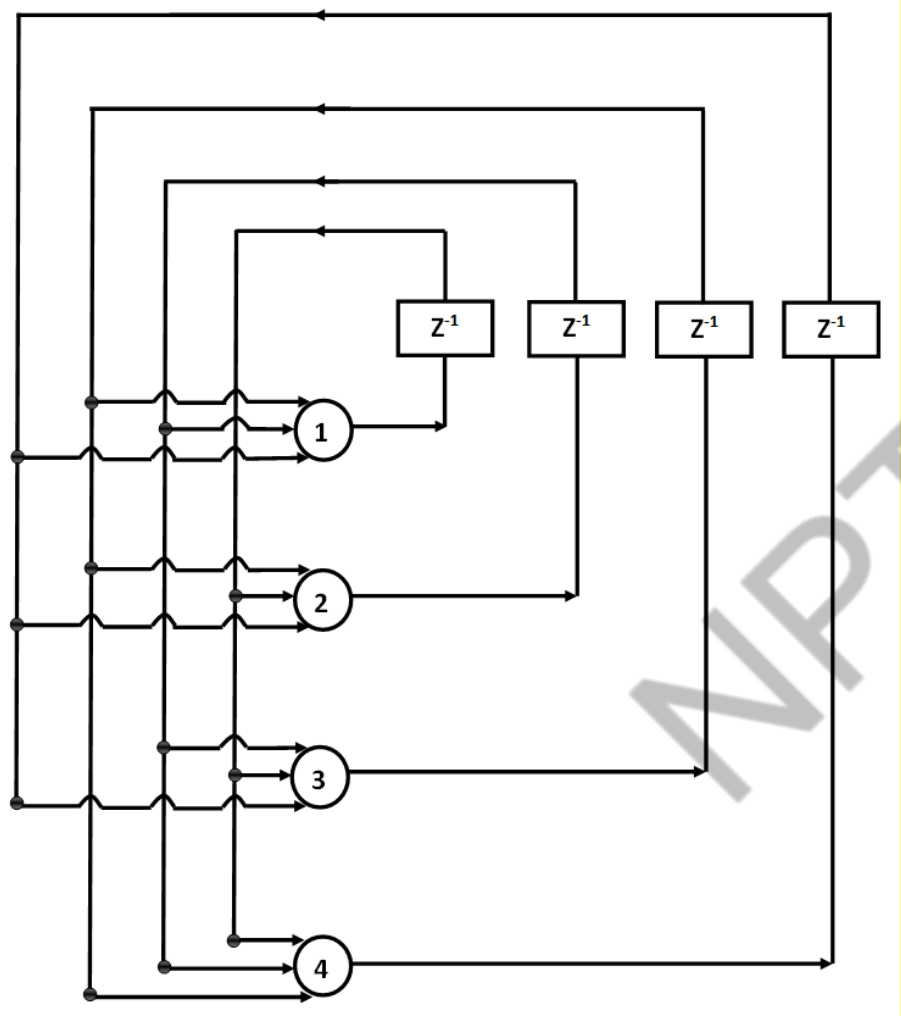
Department: Mechanical Engineering

NPTEL

Topic

Lecture 06: Some Examples of Neural Networks (Contd..)

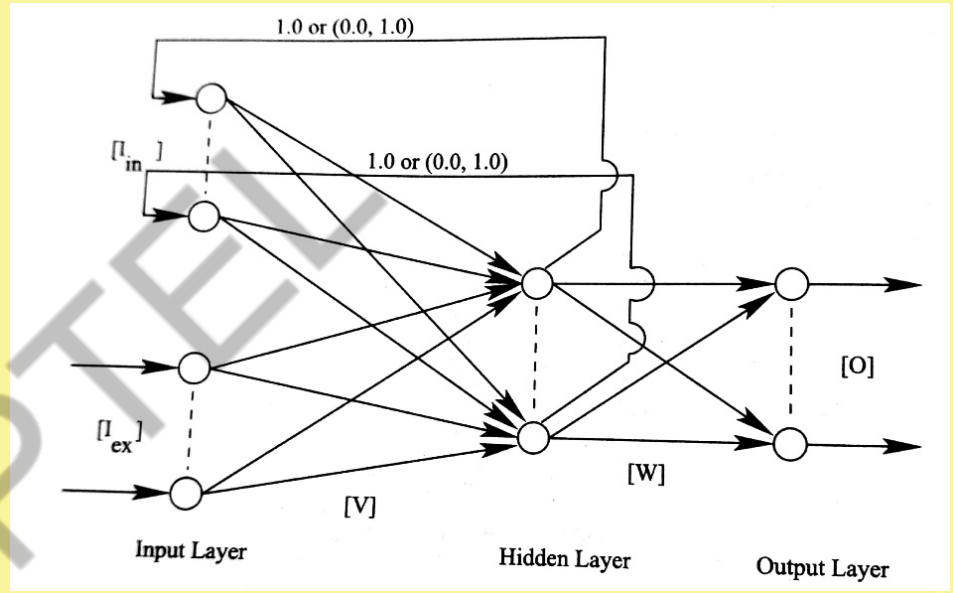
Recurrent Network of a Single Layer



NPTEL

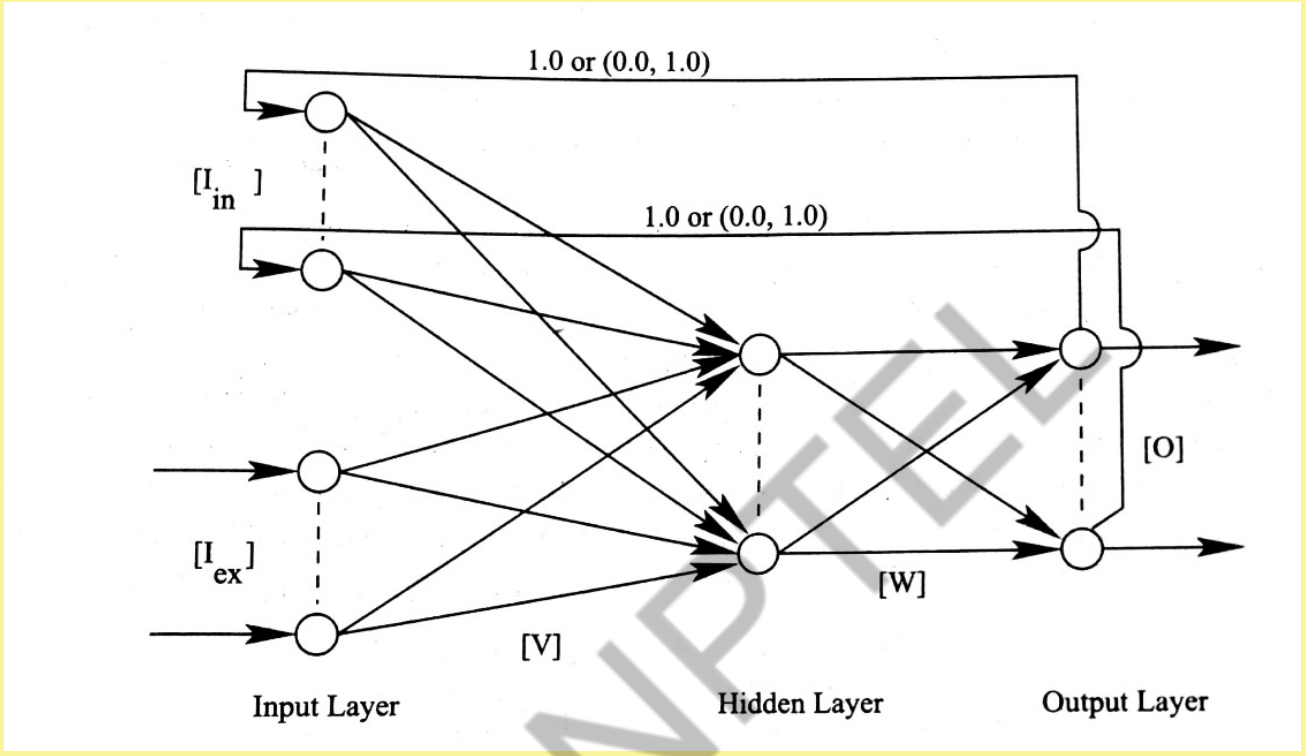
Recurrent Neural Network (RNNs)

- A recurrent neural network has both feed-forward and feed-back connections
- Information can be processed from input layer to output layer and vice-versa
- It forms a cycle or loop
- It may be preferred to MLFFNN for modeling a highly dynamic process



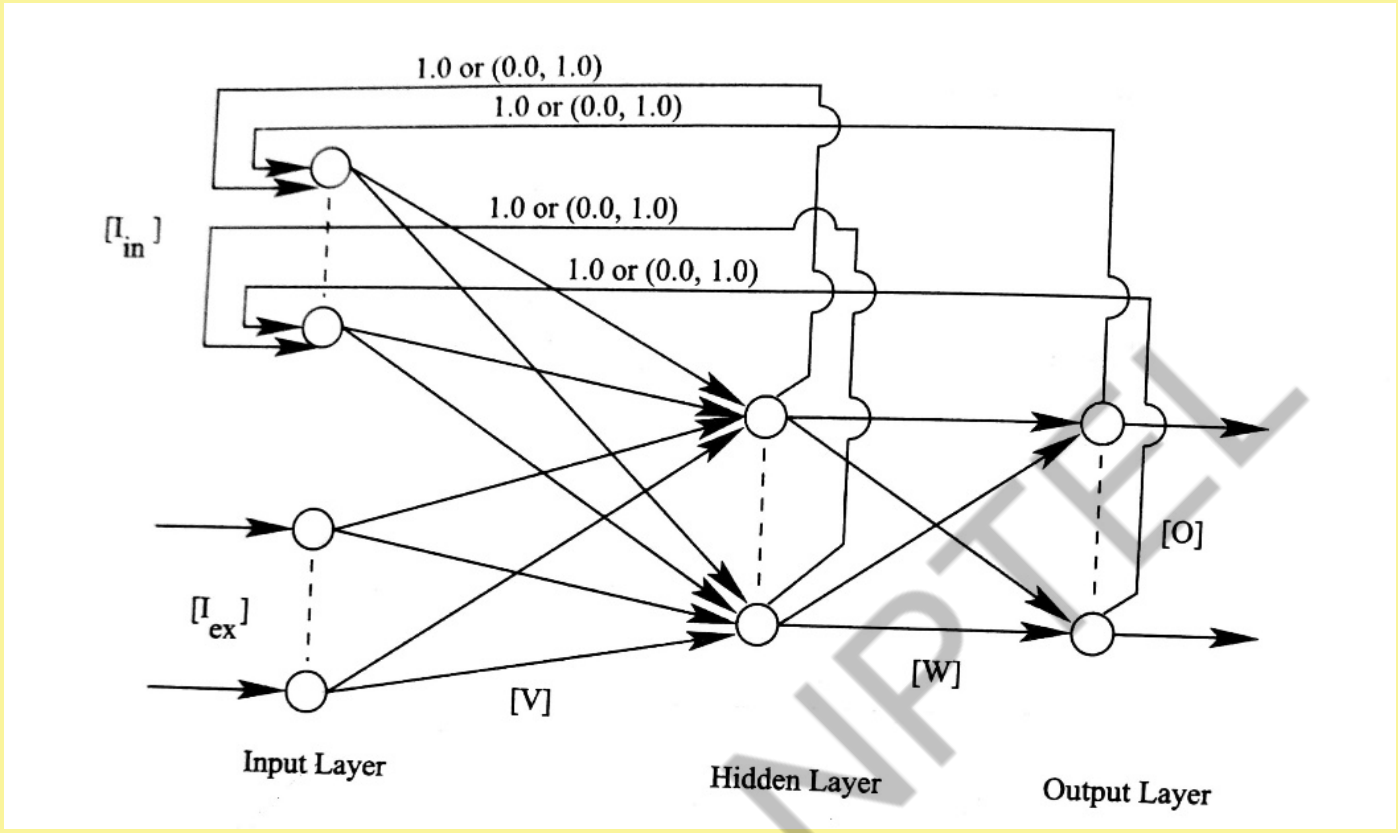
Elman Network

NPTEL



NPTEL

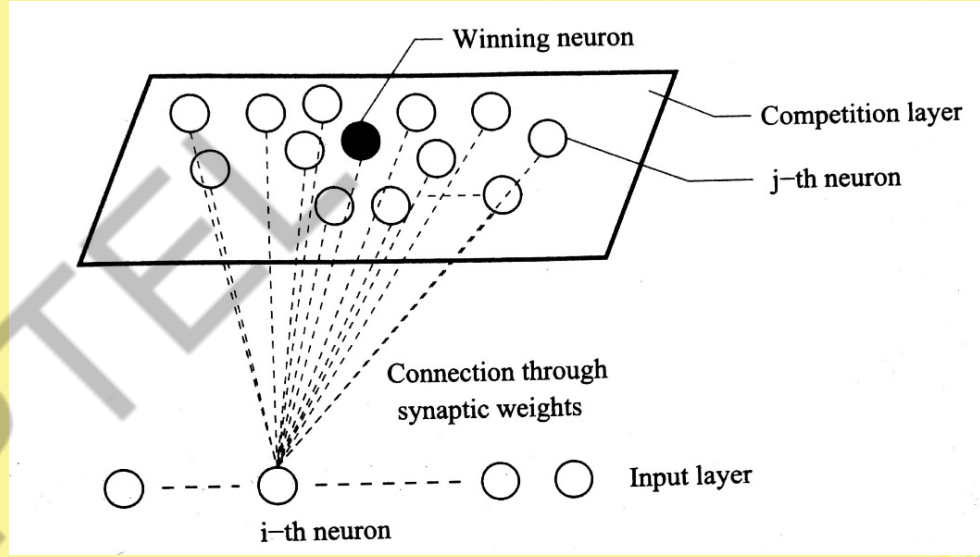
Jordan Network



Combined Elman and Jordan Network

Self-Organizing Map (SOM)/ Kohonen Network

- Proposed by T. Kohonen before 1995
- It can be used as *visualization technique* or *dimensionality reduction technique* (topology preserving tool)
- It can be used as a *clustering algorithm* also
- It woks based on *unsupervised* and *competitive* learning



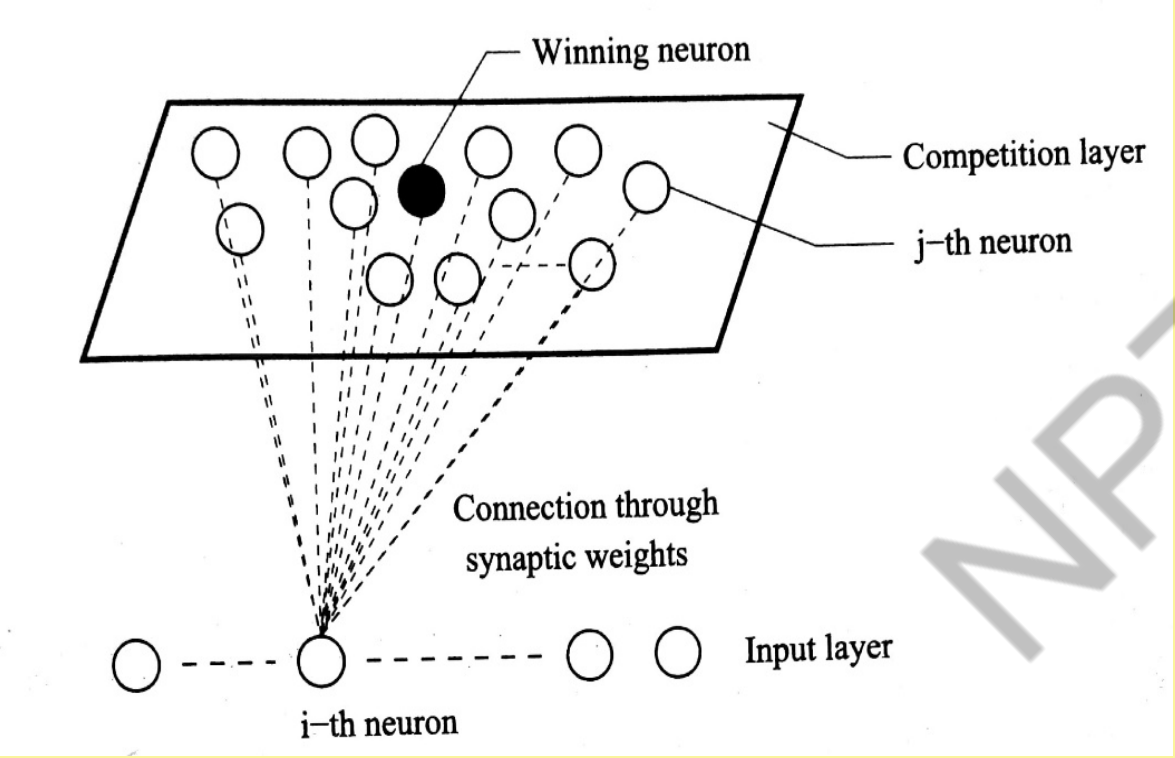
Self-Organizing Map

NPTEL

- Non-linear generalization of principal component analysis
- Consists of two layers: Input layer and competition layer
- On competition layer, there are three basic operations, namely *competition, cooperation* and *updating*

NPTEL

Self-Organizing Map (SOM)/ Kohonen Network (Cont.)



NPTEL

Competition

Let us assume that there are N points (neurons) in the input layer and each point has m dimension

$X_i = [x_{i1}, x_{i2}, \ldots, x_{im}]^T$,

where i = 1, 2, ..., N

Let the synaptic weight vector between input neuron i and neuron j lying in the competition layer is denoted by

$$W_j^i = \left[w_{j1}^i, w_{j2}^i, \ldots\ldots, w_{jm}^i \right]^T,$$

where j=1, 2, ... , N

NPTEL

Let n: neuron lying in the competition layer that has the best match with the input vector X_i ⟹ Winning neuron

Euclidean distance between n and X_i can be expressed as follows:

$n(X_i)$ = Minimum of $\sqrt{\left(X_i - W_j^i\right)^2}$

where j = 1, 2, ..., N

NPTEL

Cooperation

Surrounding a winning neuron, a neighborhood of excited neurons is defined for cooperation in order to update their synaptic weights

$$h_{j,n(x_i)}(t) = \exp\left(-\frac{d^2_{j,n(x_i)}}{2\sigma_t^2}\right),$$

where t = 0 , 1, 2, . . .

$d_{j,n(x_i)}$: Lateral distance between the winning neuron n and excited neuron j

σ_t : Standard deviation at t-th iteration

$\sigma_t = \sigma_o \exp(-t/\tau)$,

σ_o: Initial value of standard deviation

τ: Predefined number of maximum iterations

NPTEL

Updating

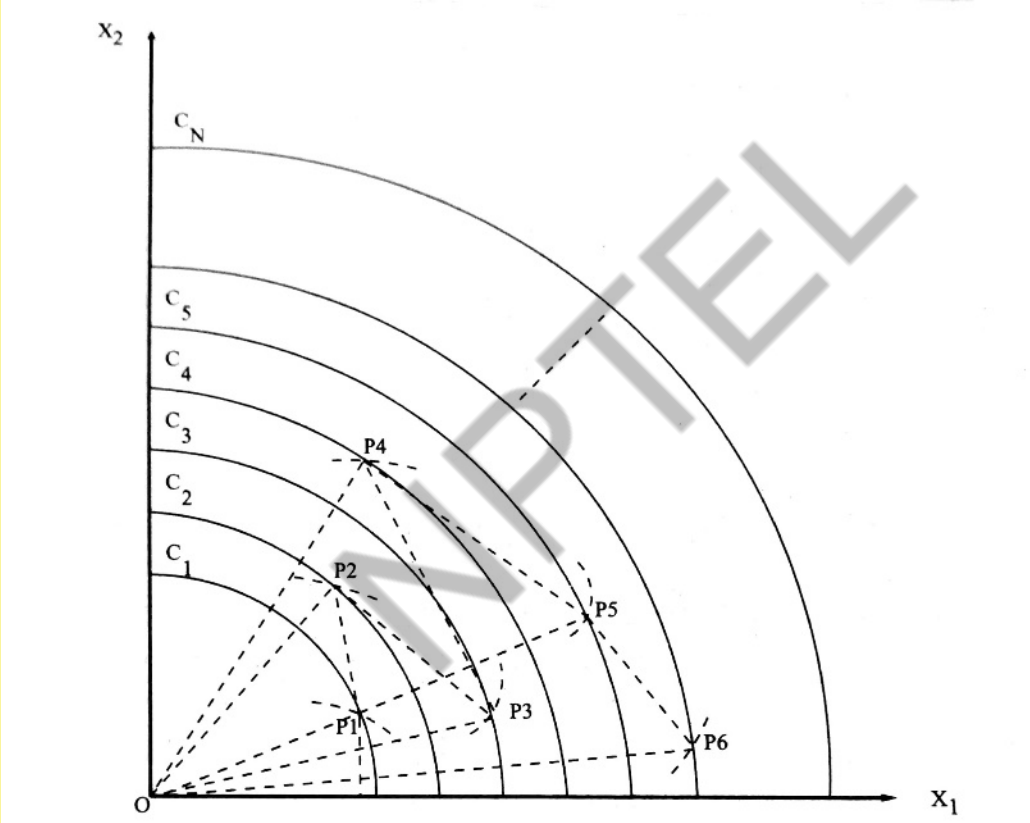
Synaptic weights of the winning neuron and excited neurons are updated as follows:

$$W_j^i(t+1) = W_j^i(t) + \eta(t) h_{j,n(x_i)}(t)\left[X_i - W_j^i(t)\right],$$

η(t): learning rate (0.0, 1.0)

NPTEL

Final Mapping

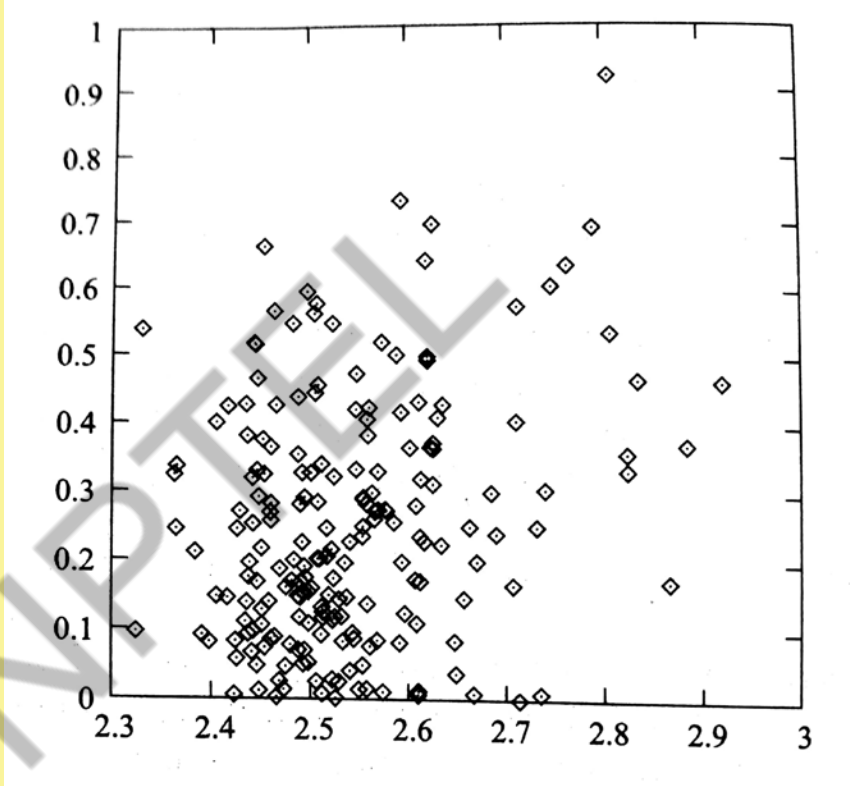


NPTEL

Simulation Results

Schaffer's first test function

$$y = 0.5 + \frac{\sin^2\sqrt{\sum_{i=1}^{4} x_i^2} - 0.5}{1.0 + 0.001\left(\sum_{i=1}^{4} x_i^2\right)^2}$$



NPTEL

Counter-Propagation Neural Network (CPNN)

- Introduced by Robert Hecht-Nielsen (1987)
- It consist of 3 layers:
  - (i) Input layer
  - (ii) Unsupervised Kohonen layer
  - (iii) Teachable output layer performing Grossberg learning (supervised)
- It contains in-star model and out-star model

  In-star model: Formed by input and Kohonen layers

  Out-star model: Formed by Kohonen and output layers

NPTEL

- CPNN is faster than BPNN and there is no chance of its weight to get trapped into local minima

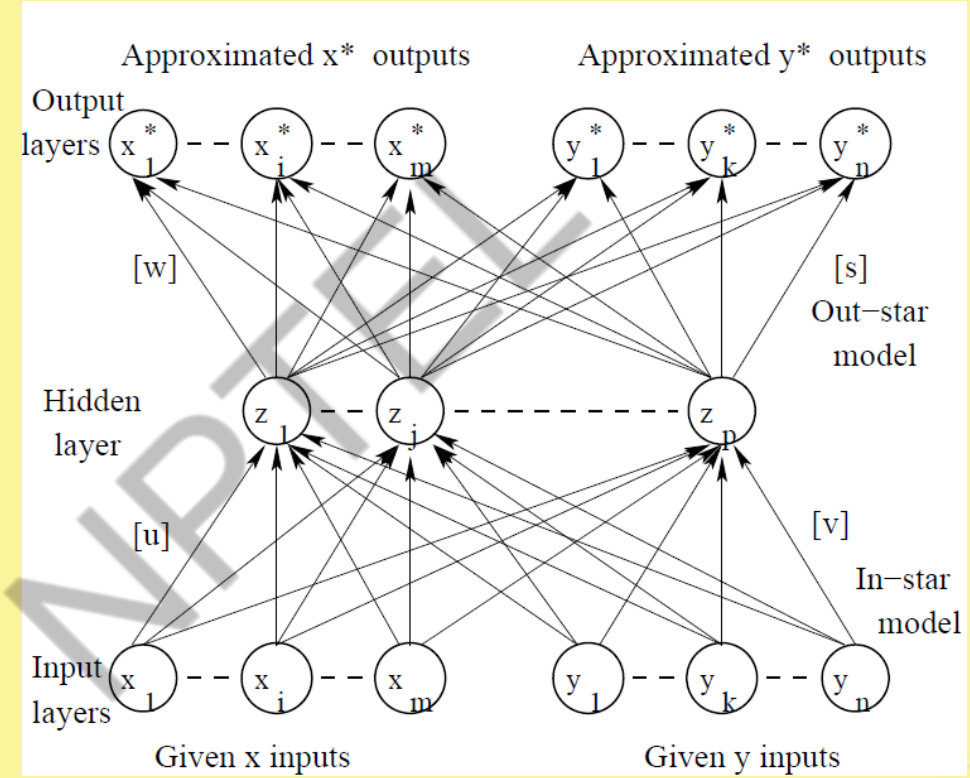
- CPNN could be inferior to BPNN in mapping applications.

- CPNN could be either a full CPNN or a forward-only CPNN.

NPTEL

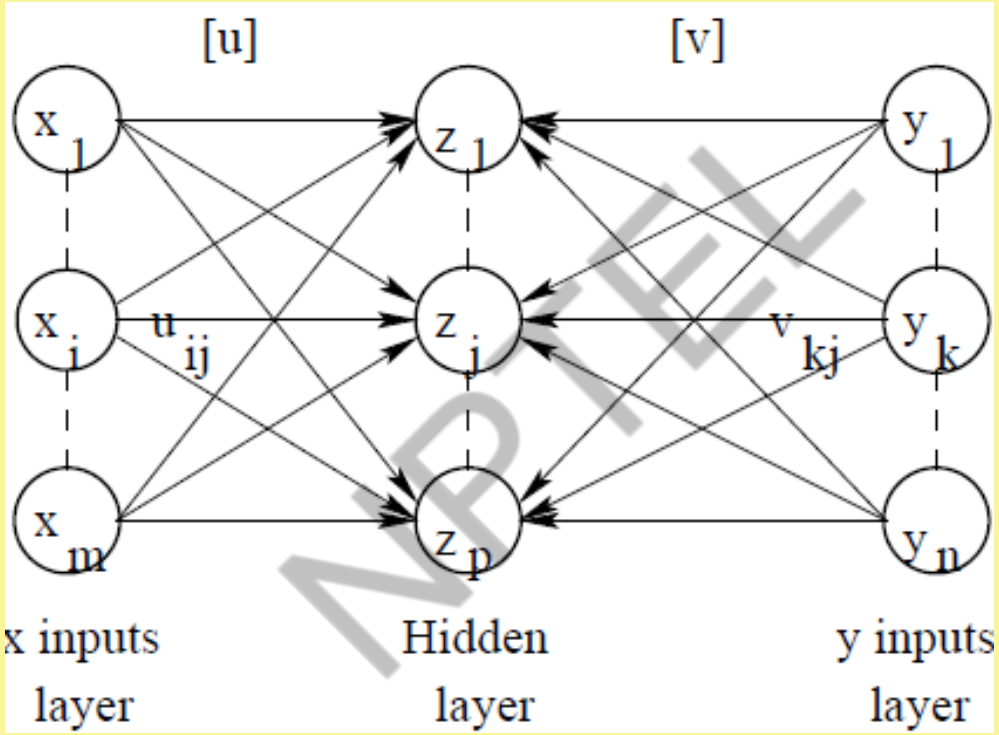
Full CPNN

- It consist of two input layers and two output layers with a common hidden layer to them.



NPTEL

Training of In-star Model



NPTEL

Training of In-star Model(continued)

- Generate the connecting weight [u] and [v] at random in the range of (0,1)
- α: learning rate between x inputs layer and hidden layer

  β: learning rate between y inputs layer and hidden layer
- Kohonen's SOM is used during the training of in-star model.

NPTEL

FREE ONLINE EDUCATION swayam

Training of In-star Model (continued)

- To decide the winner neuron, Euclidean distance values are calculated as follows:

$$d_j = \sqrt{\sum_{i=1}^{m}(x_i - u_{ij})^2 + \sum_{k=1}^{n}(y_k - v_{kj})^2}$$

where j = 1,2,……,p

u_{ij}(updated) = u_{ij}(previous)+α(x_i-u_{ij}(previous))

where i = 1,2,……,m

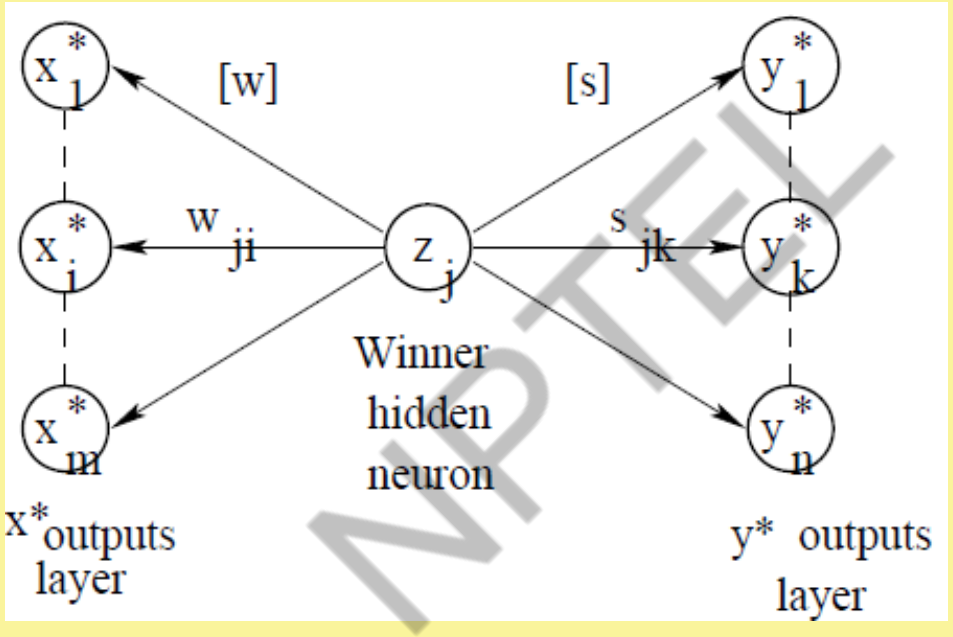
v_{kj}(updated) = v_{kj} (previous)+β(y_k-v_{kj}(previous))

where k = 1,2,……,n

❑ It completes one iteration of in-star training.

NPTEL

Training of Out-star Model



NPTEL

Training of Out-Star Model (continued)

- Say z_j be the winner hidden neuron

- Generate the connecting weights [w] and [s] at random in the range of (0,1)

- γ : learning rate between z_j and x^* output layer

  δ : learning rate between z_j and y^* output layer

NPTEL

FREE ONLINE EDUCATION swayam

Training of Out-Star Model (continued)

- Grossberg's learning is used during the training of out-star model

$w_{ji}(\text{updated}) = w_{ji}(\text{previous}) + \gamma\,(x_i - w_{ji}(\text{previous}))$

where $i = 1,2,\ldots\ldots,m$

$s_{jk}(\text{updated}) = s_{jk}(\text{previous}) + \delta\,(y_k - s_{jk}(\text{previous}))$

where $k = 1,2,\ldots\ldots,n$

$x_i^* = w_{ji}(\text{updated})$

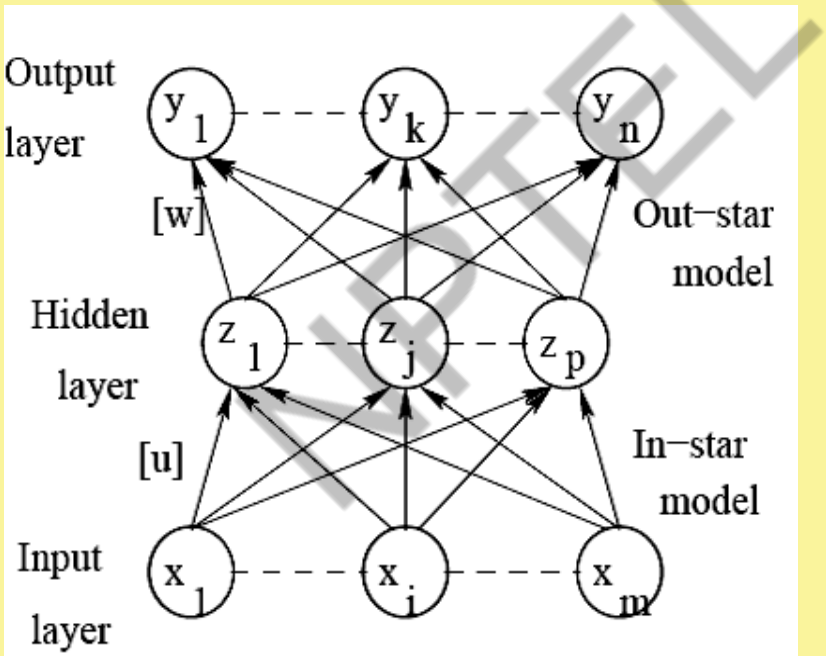
$y_k^* = s_{jk}(\text{updated})$

- It completes one iteration of out-star training.

NPTEL

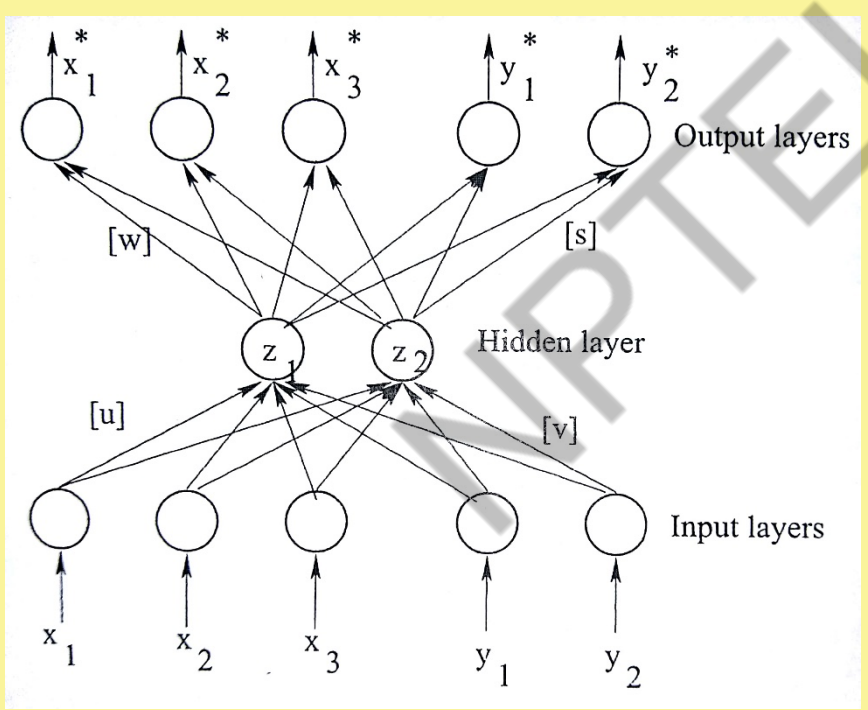
Forward-Only CPNN

- It consists of one input layer and one output layer with a common hidden layer, which performs the clustering



A Numerical Example

Let us assume that a full CPNN is to be used to model a dataset having three inputs and two outputs. One such data is represented as $(x_1 , x_2 , x_3) = (0.3,0.5,0.6)$ and $(y_1 ,y_2) = (0.3,0.4)$. Let us consider two hidden neurons only.



NPTEL

The connecting weights are initially assumed to be as follows:

$$[u]=\begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \end{bmatrix}=\begin{bmatrix} 0.2 & 0.3 \\ 0.1 & 0.6 \\ 0.8 & 0.5 \end{bmatrix}$$

$$[v]=\begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}=\begin{bmatrix} 0.4 & 0.7 \\ 0.2 & 0.3 \end{bmatrix}$$



o update the connecting weight v_{ij} between i-th neuron of nput layer and j-th neuron of hidden layer, that is,

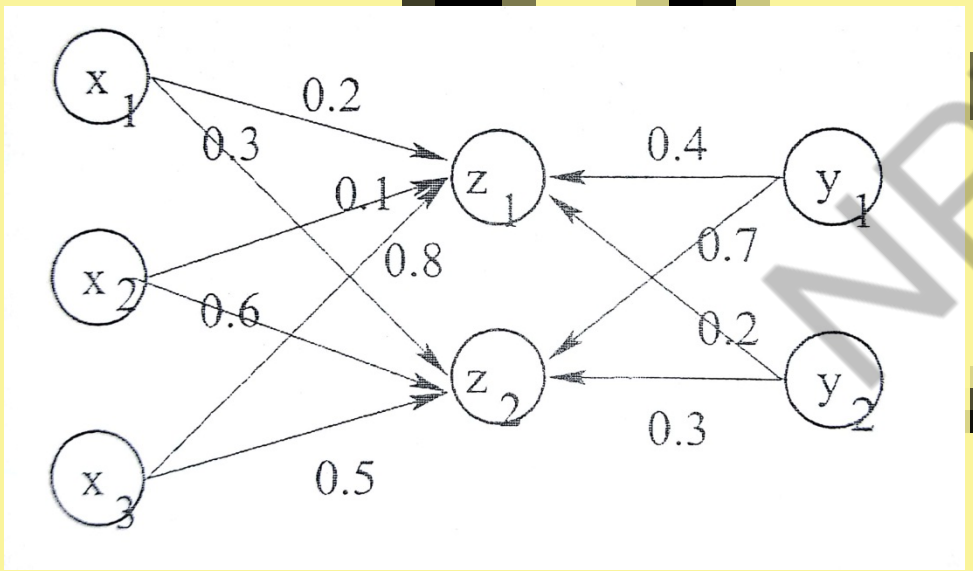
$v_{ij,\ updated} = v_{ij,\ previous} + \Delta v_{ij}$

where $\Delta v_{ij} = -\eta \left\{\frac{\partial E}{\partial v_{ij}}\right\}_{av}$

where $\left\{\frac{\partial E}{\partial v_{ij}}\right\}_{av} = \frac{1}{p}\sum_{k=1}^{p}\frac{\partial E_k}{\partial v_{ij}}$

NPTEL

V22



NPTEL

In-star model

In order to decide the winner out of two hidden neurons,

we calculate Euclidean distances as follows:

$$d_1 = \sqrt{\sum_{i=1}^{3}\left(x_i - u_{i1}\right)^2 + \sum_{k=1}^{2}\left(y_k - v_{k1}\right)^2}$$

$$= \sqrt{\left(0.3-0.2\right)^2 + \left(0.5-0.1\right)^2 + \left(0.6-0.8\right)^2 + \left(0.3-0.4\right)^2 + \left(0.4-0.2\right)^2} = 0.51$$

NPTEL

$$d_2 = \sqrt{\sum_{i=1}^{3}\left(x_i - u_{i2}\right)^2 + \sum_{k=1}^{2}\left(y_k - v_{k2}\right)^2}$$

$$= \sqrt{\left(0.3-0.3\right)^2 + \left(0.5-0.6\right)^2 + \left(0.6-0.5\right)^2 + \left(0.3-0.7\right)^2 + \left(0.4-0.3\right)^2}$$

$$= 0.44$$

As $d_2 < d_1$, z_2 is the winner.

Its connecting weights are updated as

u_{12} (updated) = u_{12} (previous) + α (x_1 - u_{12} (previous))

= 0.3 + 0.2 (0.3 - 0.3)

= 0.3

NPTEL

u_{22} (updated) = u_{22} (previous) + α (x_2 - u_{22} (previous))
= 0.6 + 0.2 (0.5 - 0.6) = 0.58

u_{32} (updated) = u_{32} (previous) + α (x_3 - u_{32} (previous))
= 0.5 + 0.2 (0.6 - 0.5) = 0.52

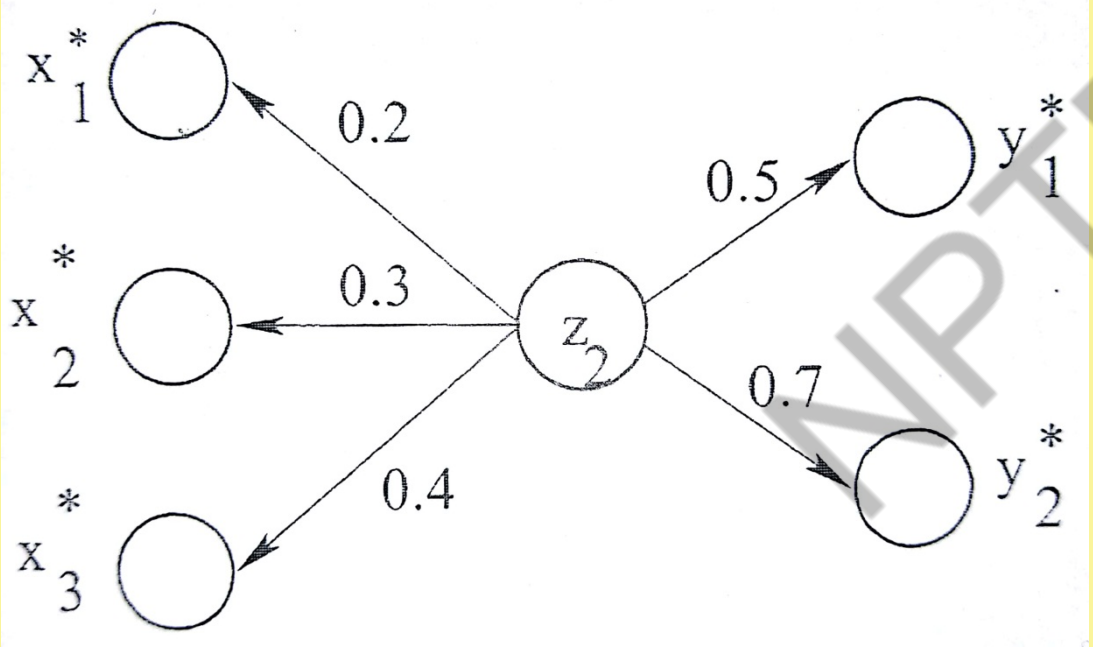
v_{12} (updated) = v_{12} (previous) + β (y_1 - v_{12} (previous))
= 0.7 + 0.3 (0.3 - 0.7) = 0.58

v_{22} (updated) = v_{22} (previous) + β (y_2 - v_{22} (previous))
= 0.3 + 0.3 (0.4 - 0.3) = 0.33

NPTEL

Out-star model

It involves only the winner neuron .



NPTEL

Out-star model

Its connecting weights are updated as

w_{21} (updated) = w_{21} (previous) + γ (x_1 - w_{21} (previous))
= 0.2 + 0.1 (0.3 - 0.2)
= 0.21

w_{22} (updated) = w_{22} (previous) + γ (x_2 - w_{22} (previous)) = 0.3 + 0.1(0.5-0.3) = 0.32

w_{23} (updated) = w_{23} (previous) + γ (x_3 - w_{23} (previous)) = 0.4 + 0.1(0.6-0.4) = 0.42

s_{21} (updated) = s_{21} (previous) + δ (y_1 - s_{21} (previous))
= 0.5 + 0.4(0.3 - 0.5) = 0.42

NPTEL

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.4 & 0.3 \\ 0.1 & 0.6 & 0.5 \end{bmatrix}';$$

NPTEL

References

References:

D.K. Pratihar Soft computing: fundamentals and applications. New Delhi, India: Narosa Publishing House, 2013.

NPTEL

Conclusion

Conclusion:

- Working principles of MLFFNN; RBFNN; RNN; SOM;

  CPNN have been discussed with suitable examples

NPTEL



FREE ONLINE EDUCATION
swayam
शिक्षित भारत, उन्नत भारत

NPTEL ONLINE CERTIFICATION COURSES

Thank you!

NPTEL