UNIT VI

CHAPTER 6

Introduction to Neural Networks

Syllabus

Artificial Neural Networks: Single Layer Neural Network, Multilayer Perceptron, Back Propagation Learning, Functional Link Artificial Neural Network, and Radial Basis Function Network, Activation functions,

Introduction to Recurrent Neural Networks and Convolutional Neural Networks

6.1 INTRODUCTION

The human brain is a highly complex and amazing processor. The most basic element of the human brain is a specific type of cell, known as Neuron, which does not regenerate. The power of human brain (mind) comes from the number of Neurons and their multiple interconnections. A brain has the ability to **Learn** to build up its own rules through what we refer to as Experience.

A **Neural-Network** is a machine that is designed to model the way in which the brain performs a particular task. Clearly, our interest is confined to an important class of **neural networks** that perform useful computation through process of **learning.** To achieve good performance, Neural networks employ a massive interconnection of simple computing cells, referred to as "Neurons" or "Processing units".

6.2 'ARTIFICIAL NEURAL NETWORK' (ANN)

Definition : Artificial Neural Network thus, we can define an 'artificial neural network' (ANN) is a massively distributed processor made up of simple processing units which store experiential knowledge and make it available for use.

The important features of 'Artificial Neural-Network (ANN)' are :

(1) Knowledge is acquired from its environment through a learning process.

(2) Interneuron-connections store the acquired knowledge.

(3) 'Artificial Neural Networks' learn by examples. In biological processes, learning involves adjustments to the connections that exist between neurons, ANNS undergo a similar changes and do the job on computers accurately all the time.

6.2.1 Concept

(1) Neural networks are those information processing systems, which are constructed and implemented to **model the human brain.**

(2) The concept that is of primary importance for a Neural Network is the ability of the network to **learn from its environment, and to improve** its performance through learning.

6.2.2 Single-Layer Network

Let us consider a Single-Layer Network

Now, the separating line is given by,

$$b + x_1 w_1 + x_2 w_2 = 0$$

If $w_2 \neq 0$, then

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

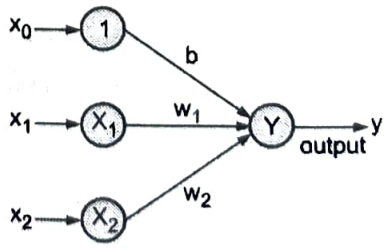


Fig. 6.2.1 : A single-layer neural-net

Thus the requirement for the positive response is : $b + x_1 w_1 + x_2 w_2 > 0$

Remarks

If the threshold value is being used, then the condition for the positive response from the output unit is

Net input received > θ (threshold value)

$\therefore y_{in} > \theta$; $\therefore x_1 w_1 + x_2 w_2 > \theta$

Then the equation of the separating line will be

$x_1 w_1 + x_2 w_2 = \theta$; $\therefore x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2}$; (with $w_2 \neq 0$)

6.3 SOLVED EXAMPLES ON ANN

Ex. 6.3.1 : For the network shown in the Fig. Ex. 6.3.1 calculate the net input to the output neuron.

Soln. :

The given neural net consists of three input neurons and one output neuron.

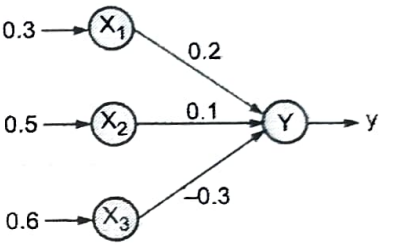


Fig. Ex. 6.3.1

The inputs are : $x_1 = 0.3$, $x_2 = 0.5$, $x_3 = 0.6$

The weights are : $w_1 = 0.2$, $w_2 = 0.1$, $w_3 = -0.3$

The net-input is given by,

$y_{in} = x_1 w_1 + x_2 w_2 + x_3 w_3 = (0.3)(0.2) + (0.5)(0.1) + (0.6)(-0.3)$

$\therefore y_{in} = 0.06 + 0.05 - 0.18 = -0.07$

Ex. 6.3.2 : Obtain the output of the neuron Y for the net-work shown in the Fig. Ex. 6.3.2, using activation functions as : (i) binary sigmoidal and (ii) bipolar sigmoidal.

Soln. :

Step (I) :

(i) The given network has three input neurons with bias and one output neuron.

These form a single – layer network.

(ii) The inputs are : $x_1 = 0.8$; $x_2 = 0.6$; $x_3 = 0.4$

and weights are $w_1 = 0.1$; $w_2 = 0.3$; $w_3 = -0.2$

and bias is b = 0.35 (its input is always 1)

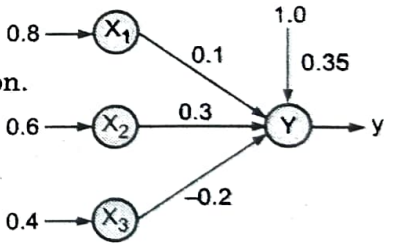


Fig. Ex. 6.3.2

Step (II) : Now the net input to the output neuron is $y_{in} = b + \sum_{i=1}^{3} x_i w_i$;

(∵ only 3 input neurons are given)

$= b + x_1 w_1 + x_2 w_2 + x_3 w_3 = 0.35 + (0.8)(0.1) + (0.6)(0.3) + (0.4)(-0.2)$

$= 0.35 + 0.08 + 0.18 - 0.08 = 0.53$

Step (III) : (i) Binary sigmoidal activation function given by,

$y = f(y_{in}) = \left(\frac{1}{1 + e^{-y_{in}}}\right) = \frac{1}{1 + e^{-0.53}} = 0.625$

(ii) bipolar sigmoidal activation functions is given by,

$$y = f(y_{in}) = \left(\frac{2}{1 + e^{-y_{in}}}\right) - 1 = \left(\frac{2}{1 + e^{-0.53}}\right) - 1 = 0.259$$

UEx. 6.3.3 (Ref. - Q. 1(d), June 14, 5 Marks)

Calculate the output of the neuron Y for the net given below. Use binary and bipolar sigmoidal activation functions :

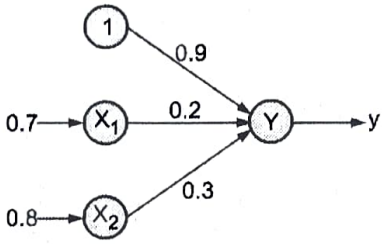


Fig. Ex. 6.3.3

Soln. :

Step (I) : (i) The given network has two input neurons with bias and one output neuron.

These form a single-layer network.

(ii) The inputs are : $x_1 = 0.7$; $x_2 = 0.8$

and weights are : $w_1 = 0.2$; $w_2 = 0.3$ and bias = $b = 0.9$ (its input is always 1)

Step (II) : Now the net input to the output neuron is $y_{in} = b + \sum_{i=1}^{2} x_i w_i$;

($\because$ only 2 input neurons are mentioned)

$$= b + (x_1 w_1 + x_2 w_2) = 0.9 + [(0.7)(0.2) + (0.8)(0.3)]$$

$$= 0.9 + 0.14 + 0.24 = 1.28 \quad \ldots(i)$$

Step (III) : (i) Binary sigmoidal function is given by

$$y = f(y_{in}) = \frac{1}{1 + e^{-y_{in}}}$$

$$= \frac{1}{1 + e^{-1.28}} = \frac{1}{1 + 0.278} = 0.7824$$

(ii) Bipolar sigmoidal activation function is,

$$y = f(y_{in}) = \left[\frac{2}{1 + e^{-y_{in}}}\right] - 1$$

$$= \left[\frac{2}{1 + e^{-1.28}}\right] - 1$$

$$= \mathbf{0.5649}$$

6.3.1 Single Layer Feed Forward Network

(1) This type of network comprises two layers, namely the **input layer** and the **output layer**.

(2) The **input layer neurons** receive' the input signals,

(3) The **output layer neurons** receive the output signals.

(4) The synaptic links carrying the weights connect every input neuron to the output neuron but not conversely.

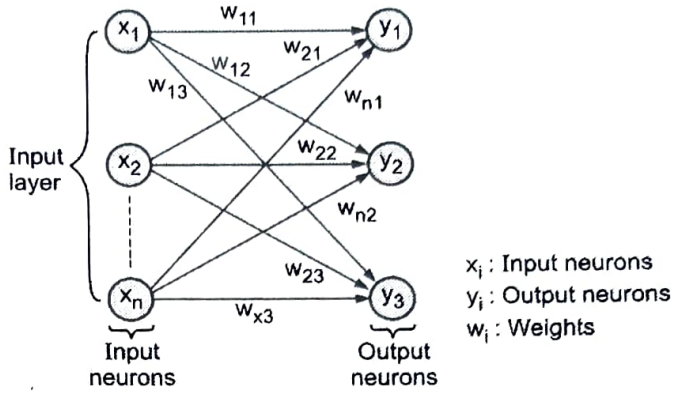


Fig. 6.3.1 : Single layer feed forward network

(5) This network is called as feed forward in type or acyclic in nature.

(6) The network is termed as single layer since the alone output layer, alone which performs computation.

(7) Even though there are two layers, it is called as single layer because of output layer.

(8) The input layer merely transmits the signals to the output layer.

We illustrate an example network as shown in Fig. 6.3.1.

6.4 MULTILAYER FEEDFORWARD NETWORK

This network is made up of multiple layers. The architecture of this class consists of one or more intermediary layers called as hidden layers, besides having an input and an output layer. The hidden layers perform intermediary computations before directing the input to the output layer. The input layer neurons are linked to the hidden layer neurons and the weights on these links are called as **input-hidden layer weights.**

Again, the hidden layer neurons are linked to the output layer neurons and the corresponding weights are referred to as **hidden-output layer weights.**

6.4.1 Perceptron Training Algorithm for Multiple output Classes

We mention below the Algorithm of perceptron training for multiple output classes :

- **Step (I) :** Initialise the weights, biases and learning rate conveniently.
- **Step (II) :** Check for stopping condition; if it is not true, perform steps 3-7.
- **Step (III) :** For each bipolar or binary training vector pair S : t, perform steps 4–6.
- **Step (IV) :** Set activation function (identity function) of each input unit i = 1 to n : xi = si
- **Step (V) :** Calculate the net input as, $yinj = bj + \sum_{i=1}^{n} xi\ wij$

Then calculate output response of each output unit j = 1 to m : To calculate the output response, apply activation formula over the net input :

$$y_j = f(y_{inj}) = \begin{cases} 1, & \text{if } y_{inj} > \theta \text{ ; } \theta \text{ is threshold valve} \\ 0, & \text{if } -\theta \le y_{inj} \le \theta \\ -1, & \text{if } y_{inj} < -\theta \end{cases}$$

- **Step (VI) :** If $t_j \neq y_j$, then make adjustment in weights and bias for j = 1 to m and i = 1 to n.

i.e. $w_{ij}\text{ (new)} = w_{ij}\text{(old)} + \alpha\, t_j\, x_i$

$b_j\text{ (new)} = b_j\text{ (old)} + \alpha\, t_j$

Otherwise we have

$w_{ij}\text{ (new)} = w_{ij}\text{ (old)}$

$b_j\text{ (new)} = b_j\text{ (old)}$

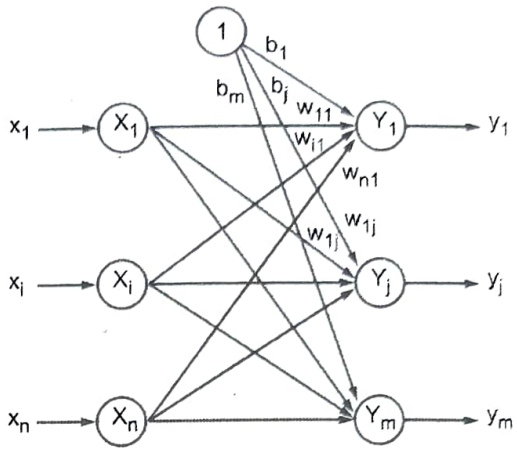


Fig. 6.4.1 : Network architecture for perceptron network for several output classes

- **Step (VII) :** If there is no change in weights then stop the training process, otherwise begin from step 3. We exhibit the architecture for the above algorithm :

6.5 PROGRAM FOR PERCEPTRON TRAINING ALGORITHM

➢ **Program 6.5.1 :** Write a program for solving linearly separable problem using Perceptron Model.

☞ **Source code**

```
%Perceptron for AND function
clear;
clc;
x=[1 1 -1 -1;1 -1 1 -1];
t=[1 -1 -1 -1];
w=[0 0];
b=0;
alpha=input('Enter Learning rate=');
theta=input('Enter Threshold value=');
con=1;
epoch=0;
while con
  con=0;
    for i=1:4
      yin=b+x(1,i)*w(1)+x(2,i)*w(2);
        if yin>theta
        y=1;
        end
          if yin <=theta & yin>=-theta
```

```
        y=0;
        end
        if yin<-theta
        y=-1;
        end
        if y-t(i)
        con=1;
          for j=1:2
            w(j)=w(j)+alpha*t(i)*x(j,i);
          end
        b=b+alpha*t(i);
        end
    end
epoch=epoch+1;
end
disp('Perceptron for AND function');
disp(' Final Weight matrix');
disp(w);
disp('Final Bias');
disp(b);
save 'result.mat','w','b';
```

☞ **Output**

```
Enter Learning rate= 0.5
Enter Threshold value=1
Perceptron for AND function
 Final Weight matrix
   1.5000    1.5000

Final Bias
  -1.5000
```

6.5.1 Perceptron Network Testing Algorithm

(I) For efficient performance of the network, it is to be trained with more data.

We mention the testing algorithm :

- **Step (I) :** The initial weights to be used are to be taken from the final weights obtained during training.
- **Step (II) :** Perform steps 3-4 for each input vector X.
- **Step (III) :** Set activations formula of the input unit.

- **Step (IV) :**Obtain the response of output unit : $y_{in} = \sum_{i=1}^{n} x_i w_i$

and $y=f(y_{in}) = \begin{cases} 1 & \text{if } y_{inj} > \theta \quad \theta \text{ is threshold} \\ 0 & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{inj} < -\theta \end{cases}$

With these steps the performance of the algorithm can be tested.

➢ **Program 6.5.2 :** Test the above trained perceptron network (Using MATLAB)

```
%This is the recall program for perceptron_AND
%Trained weight vector and bias value is saved in file 'result.mat' which is loaded here using load command
load result.mat;
x=input('Enter the test pattern');
yin=b+x(1)*w(1)+x(2)*w(2);
 if yin>theta
   y=1;
 end
 if yin <=theta & yin>=-theta
   y=0;
 end
 if yin<-theta
  y=-1;
 end
 disp('output is');
 y
```

☞ **Output**

```
Enter the test pattern
[-1 1]
output is
y =   -1
```

6.6 BACK-PROPAGATION NETWORK (BPN)

(i) Back-propagation network algorithm is applied to multilayer feed-forward networks consisting of processing elements.

(ii) The networks connected to back-propagation learning algorithm are also called as **back-propagation network (BPN).**

(iii) This algorithm provides a procedure for changing the weight in back-propagation network (BPN) to the given input pattern correctly.

(iv) The basic concept for this weight update is that where the error is propagated back to the hidden unit.

(v) In BPN, weights are calculated during the learning period of the network.

(vi) To update weights, the error must be calculated. There is no direct information of the error at the hidden layer. So we have to develop other techniques to calculate an error at the hidden layer, and this will cause minimisation of the output error.

(vii) Backpropagation is a systematic method of training multilayer artificial neural networks.

It is developed on high mathematical foundation and it has very good application potential.

(viii) Backpropagation learning rule is applicable on any feed forward network architecture.

(ix) Slow rate of convergence and local minima problems are its weaknesses.

(x) The multilayer feedforward (MLFF) network with back propagation (BP) learning is also called as 'multilayer perception' because of its similarity to perceptron networks with more than one layer.

(xi) **We carry out BPN as follows :**

(a) The feed forward of the input training pattern

(b) The back-propagation of the error and

(c) Updation of weights.

6.6.1 Architecture

A back-propagation neural network (BPN) consist of an input layer, hidden layer and an output layer. The neurons at the hidden and output layers have biases. The bias terms also act as weights. The outputs obtained could be either binary (0, 1) or bipolar (– 1, 1). Refer Fig. 6.6.1.

We Exhibit the Architecture of BPN

The terminologies used in the algorithm are as follows :

x = input training vector$(x_1, x_2, \ldots, x_n)$

t = target output vector

$(t_1, t_2, \ldots t_k, \ldots, t_m)$

α = learning rate parameter ;

v_{oj} = bias on j^{th} hidden unit,

w_{ok} = bias on k^{th} output unit ;

Z_j = hidden unit j.

x_j = input unit j, (since identity activation function is used for input layer, the input and output signals are same).

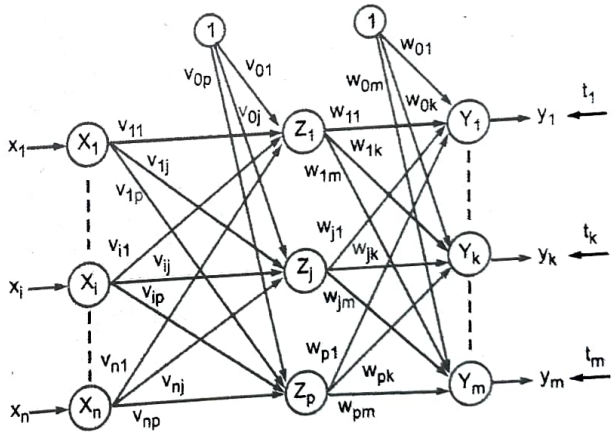


Fig. 6.6.1 : Architecture of a back-propagation network

The net input to Z_j is, $Z_{inj} = V_{oj} + \sum_{i=1}^{n} x_i v_{ij}$ and the output is, $Z_j = f(Z_{inj})$

y_k = output unit k.

The net input to y_k is, $y_{ink} = w_{ok} + \sum_{j=1}^{p} Z_j w_{jk}$ and the output is $y_k = f(y_{ink})$

δ_k = error correction weight adjustment for w_{jk}, which is back-propagated to the hidden units that feed into unit y_k. and

δ_j = error connection weight adjustment for v_{ij} and to the hidden unit Z_j.

6.6.2 Algorithm (Training)

We mention the error-back-propagation learning algorithm :

- **Step (I)** : Small random values for weights and learning rate,
- **Step (II)** : Carry on the steps 3-10 when stopping condition fails.
- **Step (III)** : Carry on steps 3-9 for each training pair.

Phase (I) : Feed-forward phase

- **Step (IV)** : Each input receives input signal xi and sends to hidden unit for i = 1 to n.
- **Step (V)** : Calculate net input

$$Z_{inj} = V_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

(To calculate output, we apply activation function Z_{inj})(binary or bipolar sigmoidal activation function) : $Z_j = f(Z_{inj})$ and send the output signal to the input of output layer units.

- **Step (VI) :** For each output y_k (k = 1 to m), calculate the net input :

$$y_{ink} = w_{ok} + \sum_{j=1}^{p} Z_j w_{jk}$$

and we apply activation function to evaluate output signal : $y_k = f(y_{ink})$

Phase II + (Back-Propagation of Error)

- **Step (VII) :** We compute error correction term for each output unit y_k (K = 1 to m)

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

Where $f'(y_{ink})$ is derivative of $f(y_{ink})$.

Now, on the basis of the calculated error correction term, update the change in weights and bias :

$$\Delta w_{jk} = \alpha \delta_k z_j \; ; \quad \Delta w_{ok} = \alpha \delta_k$$

The error δ_k is to be sent to the **hidden layer backwards.**

- **Step (VIII) :** For each hidden unit (Z_j, J = 1 to p) ;

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

and to calculate the error term : $\delta_j = \delta_{inj} \cdot f'(Z_{inj})$

Now, we update the change in weights and bias ;

$$\Delta V_{ij} = \alpha \delta_j x_i \; ; \quad \Delta V_{oj} = \alpha \delta_j$$

Phase (III) : Weight and Bias Updation

- **Step (IX) :** Each output unit (y_k, k = 1 to m) updates the bias and weights :

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} \quad ; \quad w_{ok}(new) = w_{ok}(old) + \Delta w_{ok}$$

Each hidden unit (Z_j, j = 1 to p) update its bias and weights :

$$V_{ij}(new) = V_{ij}(old) + \Delta V_{ij} \text{ and } v_{oj}(new) = V_{oj}(old) + \Delta V_{oj}$$

- **Step (X) :** Check for stopping condition.

6.6.3 Flow-chart for Back-Propagation Network Training

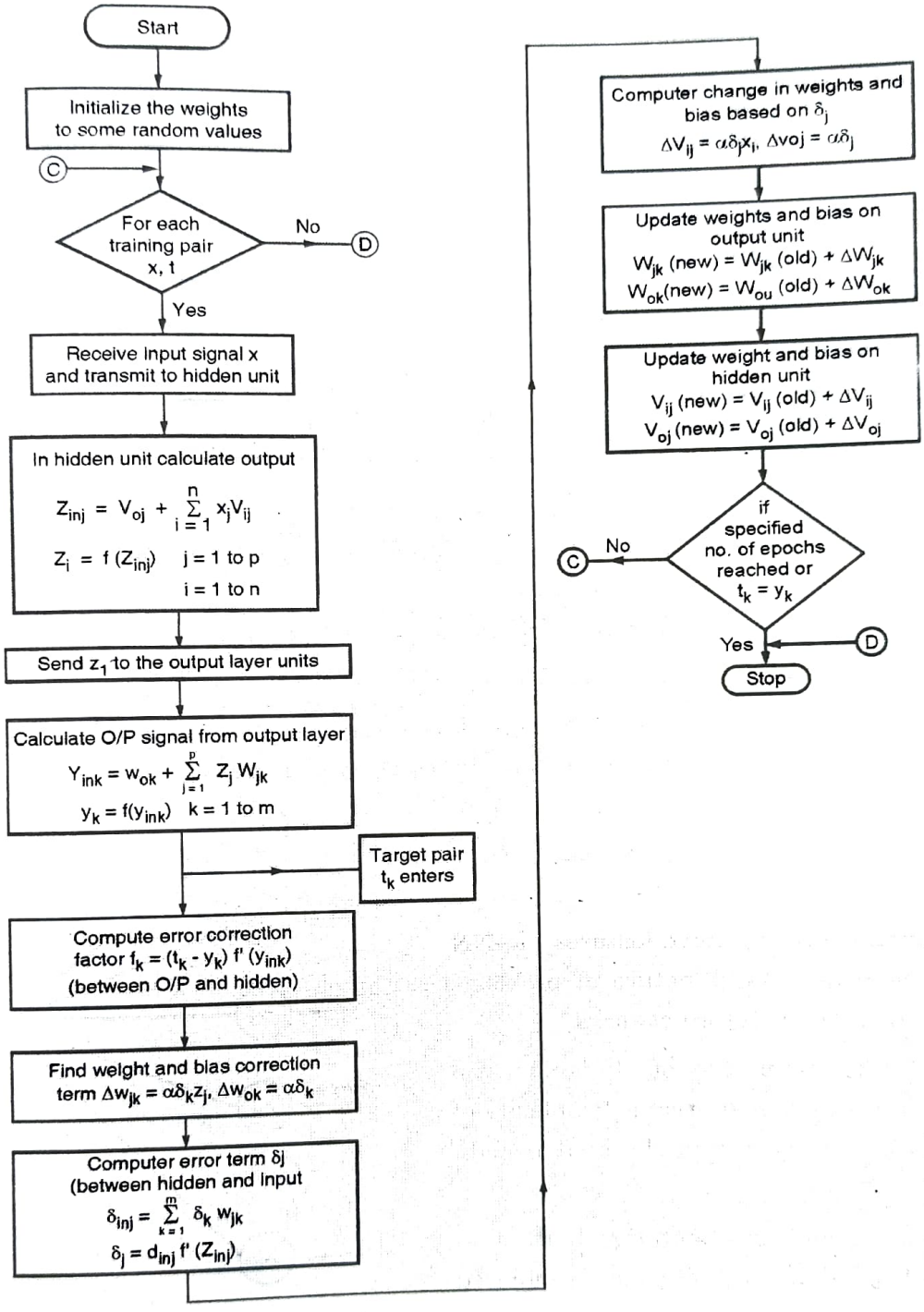


Fig. 6.6.2 : Flow-chart for Back-Propagation Network Training

6.7 LEARNING FACTORS OF BACK PROPAGATION NETWORK

(i) Learning rate α determines the size of the weight adjustments made at each iteration and hence influences the rate of convergence.

(ii) Poor choice of the rate can result in a failure in convergence.

It is convenient to keep the rate constant through all the iterations for best results. If the learning rate α is too large, the search path will oscillate and converges more slowly than a direct descent. The range of α from 10^{-3} to 0.9 is optimistic and has been used successfully for several back-propagation algorithmic experiments. If the rate is too small, the descent will progress in small steps, increasing the time to converge. Jacobs has suggested the use of adaptive coefficient where the value of the learning rate is the function of error derivative on successive updates.

6.8 FUNCTIONAL LINK ANN

- Functional link artificial neural network (FLANN) is a single layer ANN with low computational complexity which is used in different fields of application, such as system identification, pattern recognition, prediction and classification, etc.
- In all these areas of application, FALNN has been tested and found to provide superior performance compared to their multilayer perceptron (MLP) ;
- In solving classification task of data mining, the multi-layer perceptron (MLP) takes longer time and the complexity of the network increase as the number of layers increases.
- In contrast to multiple layer networks, FLANN architecture uses a single layer feed forward network.
- Using the functionally expanded features FLANN overcomes the non-linearity nature of problems and encounters in single layer networks.
- The feature like simplicity of designing the architecture and low computational complexity of the networks encourages us to choose it in data mining task.
- In short, the input representation has been enhanced and linear separability can be achieved in the extended space.

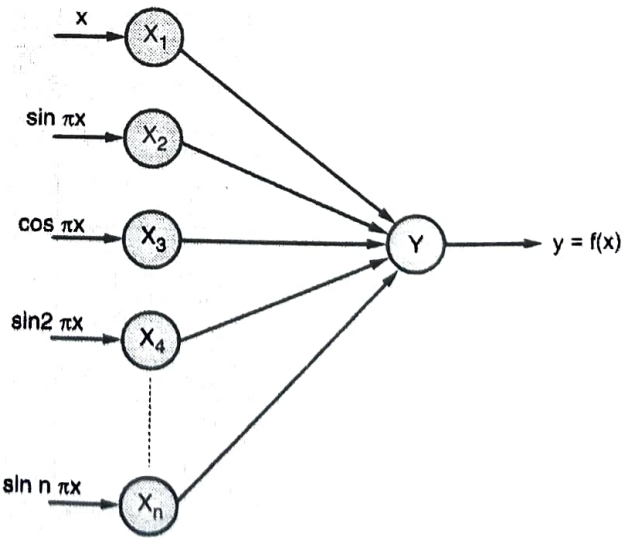


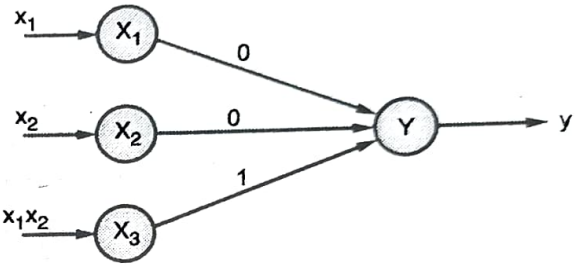
Fig. 6.8.1 : Functional link network

- We prepare a functional link model networks for learning continuous functions.
- Using (here) orthogonal basis functions such as sin πx, Cos πx, sin2 πx, Cos2 πx, etc., we can have higher order input terms.

- The most common example of linear non-separability is XOR – problem.
- The FLN can help in solving this problem. The input are

x_1	x_2	$x_1 x_2$
– 1	– 1	1
– 1	1	– 1
1	– 1	– 1
1	1	1

The functional link network consists of only one layer. As a result, the learning speed of the functional link is faster than that of BPN.



X OR - problem

Fig. 6.8.2 : XOR – Problem

6.9 RADIAL BASIS FUNCTION NETWORK

- The radial basis function (RBF) is a functional approximation neural network. It was developed by Powell.
- The network uses sigmoidal and Gaussion kernel functions.
- The response of Gaussion function is positive for all values of y and response decreases as $|y| \rightarrow O$.
- The Gaussion function is generally defined as $F(y) = e^{-y^2}$.
- The graphical representation of this function is :

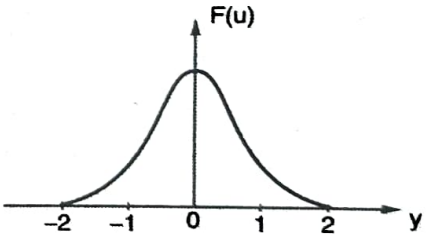


Fig. 6.9.1

- When we use Gaussion potential function then each node produces an identical output for those inputs which are at fixed **radial** distance from the centre of the kernel.
- Since the curve is symmetrical about F(y) axis, the input are radically symmetric. Hence it is named as 'Radical basis function network'
- Hence, the entire network forms a linear combination of the **nonlinear basis function.**
- We draw architectural design of RBF.

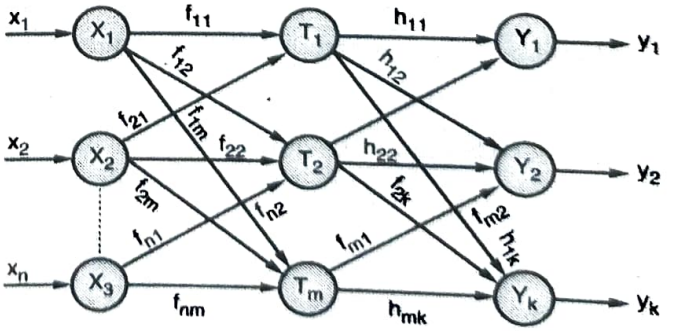


Fig. 6.9.2

- For convenience, we have chosen two layers to draw the architecture design.
- The output nodes form a **linear combination** of the basis function. This is computed by means of RBF nodes or hidden layer nodes.
- The important point is when the input falls within a small localized region of the input space formed by input stimulus in the hidden layer, it produces a significant non-zero response : This way linearity is secured.
- This network is also called as localized receptive field network.

RBF Algorithm

- We mention the RBF algorithm and it given all details of the calculations involved in the process.
- The training process begins with the hidden layer using unsupervised learning algorithm.
- The training is carried to output layer with a supervised learning algorithm.
- At the same time, for fine tuning, we apply supervised learning algorithm to both the hidden and output layers.

RBF Algorithm (Training)

- **Step 1 :** Weights of small random values to be set.
- **Step 2 :** Each input unit xi (x_i for all i = 1 to n) receives input signals and it transmits to the next hidden layer.
- **Step 3 :** To calculate the radial base function.
- **Step 4 :** Now, we select centers for the radial basis function. The centers are to be selected from the set of input vectors.

  Note that to form the linear combination of the non-linear basis, we have to choose a sufficient number of centers. That guarantees adequate sampling of the input vector space.
- **Step 5 :** Now, calculate the Output from the hidden layer unit using the formula.

$$f_j(x_i) = \frac{\exp\left[-\sum_{i=1}^{r} (x_{ji} - \bar{x}_{ji})^2\right]}{\sigma_i^2}$$

where $\bar{x}_{ji}$ is the centre of RBF unit for input variables ; σ_i the width of i^{th} RBF unit ; x_{ji} is the j^{th} variable of input pattern.

- **Step 6 :** Calculate the output of the neural network :

$$y_{net} = \sum_{i=1}^{k} h_{im} f_j(x_i) + h_o ;$$

k is the number of hidden layer nodes of RBF function. y_{net} is the output value of m^{th} node in the output layer ; w_o is the biasing term at the n^{th} output node.

- **Step 7 :** We calculate the error ; if no error, then we stop the process ; other continue process by changing the weights.

6.10 ACTIVATION FUNCTION

UQ. Explain common activation functions used in neural network. **(Ref. - Q. 1(b), May 2011, 5 Marks)**

(1) A model of the behavior of a neuron can be presented as shown in Fig. 6.10.1.

Here $x_1, x_2, ..., x_n$ are the n-inputs to the artificial neuron. $w_1, w_2, ..., w_n$ are the weights attached to the input links.

(2) **Biological** neuron receives all inputs through the dendrites, sums them and produces an output. If the sum is greater than a threshold value. The input signals are passed on to the cell body through the synapse which may accelerate or retard an arriving signal.

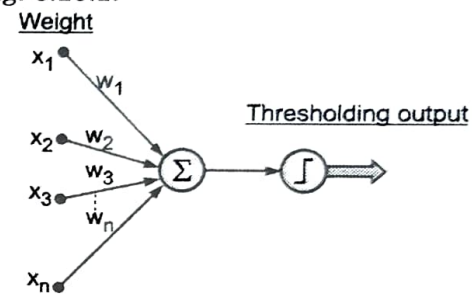


Fig. 6.10.1 : Simple model of an artificial neuron

(3) Here weights model the acceleration or retardation of the input signals. In short, weights are multiplicative factors of the inputs.

(4) The total input (say I) received by the soma (body) of the artificial neuron is

$$I = w_1 x_1 + w_2 x_2 + ... + w_n x_n = \sum_{i=1}^{n} w_i x_i \quad ...(i)$$

(5) This sum is passed on to a non-linear filter ϕ called Activation function, or Transfer function, or Squash Function which releases the outputs.

$$\text{i.e. } y = f(I) \quad ...(ii)$$

6.10.1 Activation Functions

UQ. What are the important properties of activation function used in neural networks ? **(Ref. - Q. 1(c), May 2016, 5 Marks)**

UQ. List the different activation functions used in neural network. **(Ref. - Q. 1(d), May 2017, 5 Marks)**

UQ. With mathematical list four different activation functions used in neurons. **(Ref. - Q. 1(e), May 2019, 5 Marks)**

The obtain exact output, the activation function is applied over the net input of an ANN.

There are several activation functions. Here we consider a few :

1. **Linear function :** It is defined as

$$f(x) = x \quad \text{for all } x$$

Here input = output

2. **Bipolar Step function :** The function is defined as :

$$f(x) = \begin{cases} 1, & \text{if } x \geq \theta \\ -1, & \text{if } x < \theta \end{cases}$$

Where θ is threshold value. The function is also used in single-layer nets to convert the net input to an bipolar output (+ 1, – 1).

3. **Binary step function :** The function is defined as :

$$f(x) = \begin{cases} 1, & \text{if } x \geq \theta \\ 0, & \text{if } x < \theta \end{cases}$$

This function is used in single-layer nets to convert the net input to an output that is binary (0 or 1).

4. **Sigmoidal function :** This function is used in back-propagation nets. They are of two types :

(i) Binary sigmoidal function : It is also called as unipolar sigmoid function or a logistic sigmoid function, and defined as

$f(x) = \frac{1}{1+e^{-\lambda x}}$, where λ is steepness parameter

The derivative of f(x) is :

$$f'(x) = \frac{-1}{[1+e^{-\lambda x}]^2} \cdot (-\lambda e^{-\lambda x}) = \frac{\lambda (e^{-\lambda x},)}{[1+e^{-\lambda x}]^2} = \frac{\lambda [1+e^{-\lambda x}-1]}{[1+e^{-\lambda x}]^2}$$

$$= \lambda \left\{ \frac{1}{(1+e^{-\lambda x})} - \frac{1}{(1+e^{-\lambda x})^2} \right\} = \lambda \left[\frac{1}{(1+e^{-\lambda x})} \left\{ 1 - \frac{1}{(1+e^{-\lambda x})} \right\} \right]$$

$$= \lambda [f(x)(1-f(x))] = \lambda f(x)[1-f(x)]$$

Range of this sigmoid function is 0 to 1 [$\because$ denominator is always greater than or equal to 1]

(ii) Bipolar sigmoid function :

The function is given by

$$f(x) = \frac{2}{1+e^{-\lambda x}} - 1 = \frac{2-1-e^{-\lambda x}}{1+e^{-\lambda x}} = \frac{1-e^{-\lambda x}}{1+e^{-\lambda x}}$$

where again, λ is steepress parameter and the range is between – 1 and + 1.

Derivative of f(x) is :

$$f'(x) = \frac{[1+e^{-\lambda x}][\lambda e^{-\lambda x}] - (1-e^{-\lambda x})(-\lambda e^{-\lambda x})}{(1+e^{-\lambda x})^2}$$

$$= \frac{\lambda e^{-\lambda x}[1+e^{-\lambda x}+1-e^{-\lambda x}]}{(1+e^{-\lambda x})^2} = \frac{2\lambda e^{-\lambda x}}{(1+e^{-\lambda x})^2} = \frac{\lambda}{2}\left[\frac{4e^{-\lambda x}}{(1+e^{-\lambda x})^2}\right]$$

$$= \frac{\lambda}{2}\left[\frac{(1+e^{-\lambda x})^2-(1-e^{-\lambda x})^2}{(1+e^{-\lambda x})^2}\right] = \frac{\lambda}{2}\left[1-\left(\frac{1-e^{-\lambda x}}{1+e^{-\lambda x}}\right)^2\right] = \frac{\lambda}{2}[1-f(x)^2]$$

$$= \frac{\lambda}{2}[(1+f(x))(1-f(x))] \qquad [\because a^2-b^2 = (a+b)(a-b)]$$

☞ **Remark**

(1) Here, $f(x) = \dfrac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}} = \tanh\left(\dfrac{\lambda x}{2}\right)$

$$\therefore \quad f'(x) = \frac{\lambda}{2} \operatorname{sech}^2\left(\frac{x}{2}\right)$$

$$= \frac{\lambda}{2}\left[1 - \tanh^2\left(\frac{x}{2}\right)\right] = \frac{\lambda}{2}\left[\left(1 + \tanh\frac{x}{2}\right)\left(1 - \tanh\frac{x}{2}\right)\right]$$

$$f'(x) = \frac{\lambda}{2}\left[(1 + f(x))\,(1 - f(x))\right]$$

(2) Sigmoid functions are so-called because their graphs are "S-shaped".

(i) Simple sigmoids defined to be odd, are monotone functions of one variable,

(ii) Hyperbolic sigmoids are subset of simple sigmoids and natural generalisation of the hyperbolic tangent function.

(3) **Ramp function**

It is defined as $f(x) = \begin{cases} 1, & \text{if } x > 1 \\ x, & 0 \le x \le 1 \\ 0, & x < 0 \end{cases}$

Graphs of activation functions

(1) Linear function

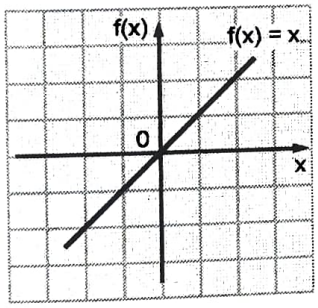


Fig. 6.10.2

(2)

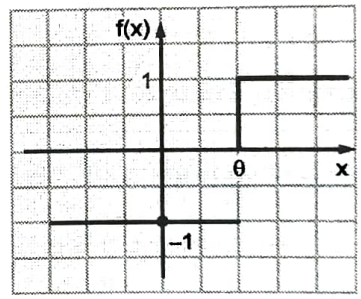


Fig. 6.10.3

(3) **Binary-step function**

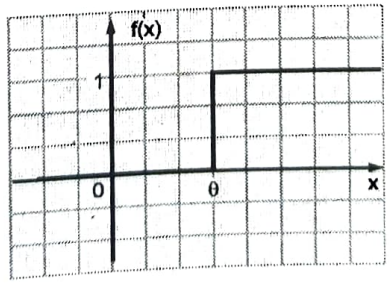


Fig. 6.10.4

(4)

(i) Binary sigmoidal function

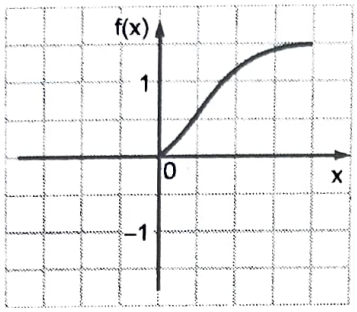


Fig. 6.10.5(i)

(ii) Bipolar sigmoidal function

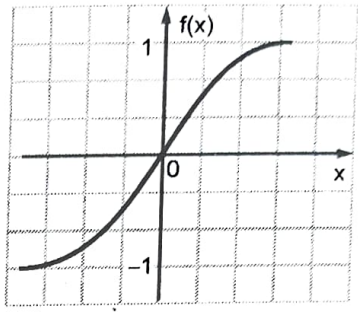


Fig. 6.10.5(ii)

(5) Ramp function

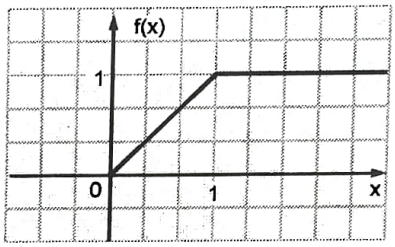


Fig. 6.10.6

6.10.2 Illustrative Examples for Logistic Function

Ex. 6.10.1 : Logistic function is given by, $F(v) = \frac{1}{1 + \exp(-av)}$

Show that the derivative of f(v) w.r.t. v is given by, $\frac{df}{dv} = af(v)\,[1 - f(v)]$

What is the value of this derivative at the origin ?

Soln. :

▶ **Step (I) :** We have, $f(v) = \frac{1}{1 + \exp(-av)}$

Different w.r.t. v ; we get

$$f(v) = \frac{-1}{[1 + \exp(-av)]^2}\,[\exp(-av)(-a)] = \frac{a \exp(-av)}{[1 + \exp(-av)]^2}$$

$$= \frac{a}{[1 + \exp(-av)]}\,\frac{\exp(-av)}{[1 + \exp(-av)]} = a\,f(v)\left[\frac{\exp(-av)}{[1 + \exp(-av)]}\right]$$

$$= a\,f(v)\left[1 - 1 - \frac{\exp(-av)}{1 + \exp(-av)}\right] = a\,f(v)\left[1 - \left\{1 + \frac{\exp(-av)}{1 + \exp(-av)}\right\}\right]$$

$$= a\,f(v)\left[1 - \left\{\frac{1 + \exp(-av) - \exp(-av)}{1 + \exp(-av)}\right\}\right] \qquad = a\,f(v)\left[1 - \left\{\frac{1}{1 + \exp(-av)}\right\}\right]$$

$\therefore f'(v) = a\, f(v)\,[1 - f(v)]$

Step (II) : As $V \to 0$; $\exp(-av) = \exp(0) = 1$

$$\therefore f(v) = \frac{1}{1+1} = \frac{1}{2}$$

$\therefore f'(v)$ at $v = 0$ is,

$$= a \cdot \frac{1}{2}\left[1 - \frac{1}{2}\right] = \frac{9}{4}$$

Ex. 6.10.2 : An odd sigmoid function is given by $f(v) = \dfrac{1 - \exp(-av)}{1 + \exp(-av)} = \tan h\left(\dfrac{av}{2}\right)$.

Show that the derivative of f(v) w.r.t. V is given by, $\dfrac{df}{dv} = \dfrac{a}{2}[1 - f^2(v)]$.

What is the valve of this derivative at the origin ? Suppose that the slope parameter a is made infinitely large. What is the resulting form of f(v).

☑ **Soln. :**

Step (I) : For convenience we take, $f(v) = \tan h\left(\dfrac{av}{2}\right)$

Differentiating w.r.t. V;

$$\frac{df}{dv} = \operatorname{sech}^2\left(\frac{av}{2}\right) \cdot \frac{a}{2} = \frac{a}{2}\left[1 - \tan h^2\left(\frac{av}{2}\right)\right]$$

$$\left[\because \operatorname{sech}^2 x = 1 - \tanh^2 x\right]$$

$$= \frac{a}{2}[1 - f^2(v)]$$

$$\therefore \frac{df}{dv} = \frac{a}{2}[1 - f^2(v)] \qquad \ldots(i)$$

Step (II) : We have $f(v) = \tanh\left(\dfrac{av}{2}\right)$

At $v = 0$, $f(0) = \tan h(0) = 0$. $\therefore$ At origin, from Equation (i), $\dfrac{df}{dv} = \dfrac{a}{2}[1 - 0] = \dfrac{a}{2}$

As $a \to \infty$, $\tan h(\infty) = 1$. $\therefore$ As $a \to \infty$, $f(v) \to 1$.

Ex. 6.10.3 : The algebraic sigmoid function is given by, $f(v) = \dfrac{v}{\sqrt{1 + v^2}}$

Show that the derivative of f(v) w.r.t. v is given by, $\dfrac{df}{dv} = \dfrac{\phi^3(v)}{v^3}$

What is the value of this derivative at origin ?

☑ **Soln. :**

Step (I) :

Differentiating $f(v) = \dfrac{v}{\sqrt{1 + v^2}}$; we get

$$f'(v) = \left[\frac{\sqrt{1+v^2} \cdot 1 - v \cdot \frac{1 \cdot 2v}{2\sqrt{1+v^2}}}{(1+v^2)}\right] = \left[\frac{\sqrt{1+v^2} - \frac{v^2}{\sqrt{1+v^2}}}{(1+v^2)}\right] = \left[\frac{(1+v^2) - v^2}{(1+v^2)^{3/2}}\right]$$

$$\therefore f'(v) = \frac{1}{(1+v^2)^{3/2}} \quad ...(i)$$

$$\therefore f'(v) = \frac{1}{v^3}\left[\frac{v^3}{(1+v^2)^{3/2}}\right] = \frac{1}{v^3}\left[\frac{v}{(1+v^2)}\right]^3$$

$$= \frac{1}{v^3}[f(v)]^3 = \frac{f^3(v)}{v^3} \quad ...(ii)$$

▶ **Step (II) :** As $v \to 0$ $f'(v) = \frac{1}{(1+v^2)^{3/2}} \to 1$.

$\therefore$ At origin, $f'(v) = 1$.

6.11 RECURRENT NEURAL NETWORK (RNN)

- Recurrent neural network is a class of artificial neural network where connections between nodes form a directed graph along a temporal sequence. This makes it to show temporal dynamic behaviour.
- It uses sequential data or time series data. These data are commonly used for ordinal or temporal problems, such as languages translation, natural language processing, speech recognition and image captioning, they are incorporated into popular applications such as voice search and Google translation.
- Like CNNs, recurrent neural networks utilise training data by learning. They are distinguished by their 'memory' as they take information from prior inputs to influence the current input and output.
- While traditional deep neural networks assume that inputs and outputs are independent of each other. The output of recurrent neural networks depends on the prior elements within the sequence, while future events would be also helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions.

6.12 CONVOLUTIONAL NETWORKS

GQ. Explain convolutional networks.

Convolutional network is a **special class** of **multilayer perceptrons,** designed to recognise two-dimensional shapes with a high degree of accuracy to skewing, scaling and to translation and other forms of distortion.

This task is achieved by a supervised manner which includes the following forms of constraints :

(1) Each neuron extracts local feature from the previous layer. Convolution layers apply a convolution operation to the input passing the result to the next layer. A convolution converts all the pixels in it's receptive field into a single value. The final output of the convolutional layer is a vector.

- Convolutional layers are the major building blocks used in convolution neural networks.

- In convolution layer, a neuron is only connected to a local area of input.
- A convolution layer contains units whose receptive fields cover a patch of receptive layer.
- The convolution layer is the core building block of a convolution network.

(2) Each computational layer is composed of feature maps in the form of a plane in which each individual neuron share the same synaptic weights. It has the following beneficial effects :

(i) Shift invariance through the use of **convolution** with a **kernel** of small size (and using sigmoid functions)

(ii) Reduction in the number of free parameters.

(3) **Subsampling :** Each convolutional layer performs local averaging and subsampling. It reduces the sensivity of the feature maps distortion.

- In convolutional network, all weights in all layers are learned through training. The network learns to extract it's own features automatically.
- In deep learning, a convolutional neural network (CNN) is an artificial neural network, applied to analyse visual imagery. They are also called as shift invariant or space invariant artificial neural networks (SIANN). Convolutional neural networks are equivariant and not invariant to translation.
- They have applications in image and video recognition, image segmentation, image classification, medical image analysis, image and video recognition, natural language processing brain-computer interfaces, and financial time series.
- CNNs are regularised versions of multilayer perceptronsy which are fully connected layers, i.e. each neuron in one layer is connected to all neurons in the next layer.

These days, deep learning is the threshold of many advanced technological applications, from self-driving cars to art and music. Deep learning enables the computer to build complex concepts from simpler concepts. Very soon we shall see that deep learning will be extended to applications that enable machines to thinks on their own.

6.13 CONVOLUTION NEURAL NETWORK (CNN) ARCHITECTURE

GQ. Explain CNN architecture.

- A **Convolution Neural Network** (CNN) is a feed-forward neural network (FNN). So far CNNs have established extraordinary performance in image search services, voice recognition and natural language processing (NLP).
- We have already seen that a regular multilayer perceptron gives good result for small images. But it breaks down for larger images.
- For example, if the first layer has 1000 neurons, then there will be 10 million connections.
- CNNs solve this problem using partially connected layers. CNN has fewer parameters than a deep neural network (DNN), hence it requires less training data.

In addition, CNN can detect any particular feature anywhere on the image, but a DNN can detect it only in that particular location.

- Since images have generally repetitive features, CNNs can generalise much better than DNNs for image processing tasks such as classification.
- A CNNs architecture has prior knowledge of how pixels are organised.
- Lower layers identify features in small areas of the images, while higher layers combine features of lower-layer into larger features. In case of DNNs, this doesn't work.
- Thus, in short, CNN is a class of neural networks that specialises in processing data that has a grid-like topology, such as an image.
- Each neuron works in it's own receptive field and is connected to other neurons in a way that they cover entire visual field.
- A CNN design begins with feature extraction and finishes with classification.

☞ **DNN Neural network**

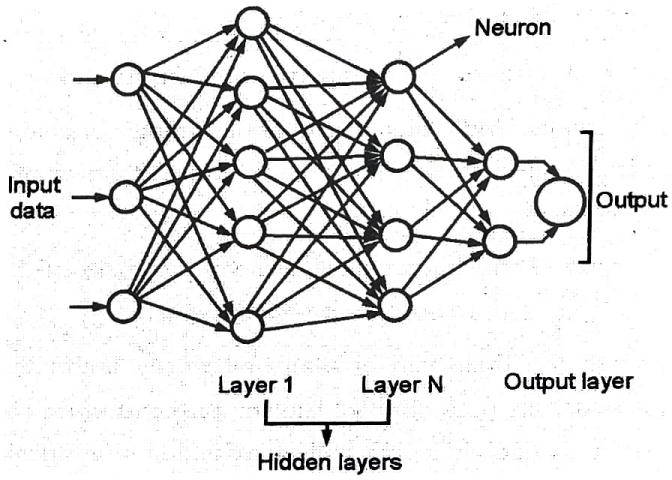


Fig. 6.13.1

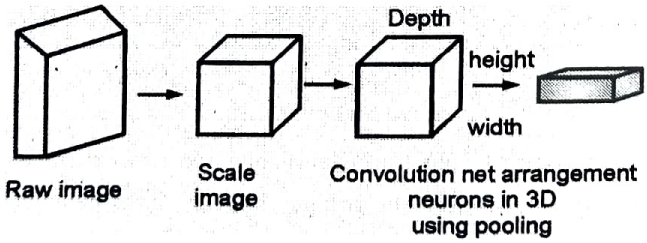


Fig. 6.13.2

In Fig. 6.13.1, there is DNN neural network. On the right, a convolution net arranges it's neurons in 3-dimensions, as seen in one of the layers.

6.14 CONVOLUTIONAL NETWORKS

GQ. Explain convolutional networks.

Convolutional network is a **special class** of **multilayer perceptrons,** designed to recognise two-dimensional shapes with a high degree of accuracy to skewing, scaling and to translation and other forms of distortion.

This task is achieved by a supervised manner which includes the following forms of constraints :

(1) Each neuron extracts local feature from the previous layer. Convolution layers apply a convolution operation to the input passing the result to the next layer. A convolution converts all the pixels in it's receptive field into a single value. The final output of the convolutional layer is a vector.

- Convolutional layers are the major building blocks used in convolution neural networks.
- In convolution layer, a neuron is only connected to a local area of input.
- A convolution layer contains units whose receptive fields cover a patch of receptive layer.
- The convolution layer is the core building block of a convolution network.

(2) Each computational layer is composed of feature maps in the form of a plane in which each individual neuron share the same synaptic weights. It has the following beneficial effects :

(i) Shift invariance through the use of **convolution** with a **kernel** of small size (and using sigmoid functions)

(ii) Reduction in the number of free parameters.

(3) **Subsampling :** Each convolutional layer performs local averaging and subsampling. It reduces the sensivity of the feature maps distortion.

- In convolutional network, all weights in all layers are learned through training. The network learns to extract it's own features automatically.
- In deep learning, a convolutional neural network (CNN) is an artificial neural network, applied to analyse visual imagery. They are also called as shift invariant or space invariant artificial neural networks (SIANN). Convolutional neural networks are equivariant and not invariant to translation.
- They have applications in image and video recognition, image segmentation, image classification, medical image analysis, image and video recognition, natural language processing brain-computer interfaces, and financial time series.
- CNNs are regularised versions of multilayer perceptronsy which are fully connected layers, i.e. each neuron in one layer is connected to all neurons in the next layer.

These days, deep learning is the threshold of many advanced technological applications, from self-driving cars to art and music. Deep learning enables the computer to build complex concepts from simpler concepts. Very soon we shall see that deep learning will be extended to applications that enable machines to thinks on their own.

6.15 CONVOLUTION NEURAL NETWORK (CNN) ARCHITECTURE

GQ. Explain CNN architecture.

- **A Convolution Neural Network** (CNN) is a feed-forward neural network (FNN). So far CNNs have established extraordinary performance in image search services, voice recognition and natural language processing (NLP).
- We have already seen that a regular multilayer perceptron gives good result for small images. But it breaks down for larger images.
- For example, if the first layer has 1000 neurons, then there will be 10 million connections.
- CNNs solve this problem using partially connected layers. CNN has fewer parameters than a deep neural network (DNN), hence it requires less training data.
- In addition, CNN can detect any particular feature anywhere on the image, but a DNN can detect it only in that particular location.
- Since images have generally repetitive features, CNNs can generalise much better than DNNs for image processing tasks such as classification.
- A CNNs architecture has prior knowledge of how pixels are organised.
- Lower layers identify features in small areas of the images, while higher layers combine features of lower-layer into larger features. In case of DNNs, this doesn't work.
- Thus, in short, CNN is a class of neural networks that specialises in processing data that has a grid-like topology, such as an image.
- Each neuron works in it's own receptive field and is connected to other neurons in a way that they cover entire visual field.

A CNN design begins with feature extraction and finishes with classification.

☞ **DNN Neural network**

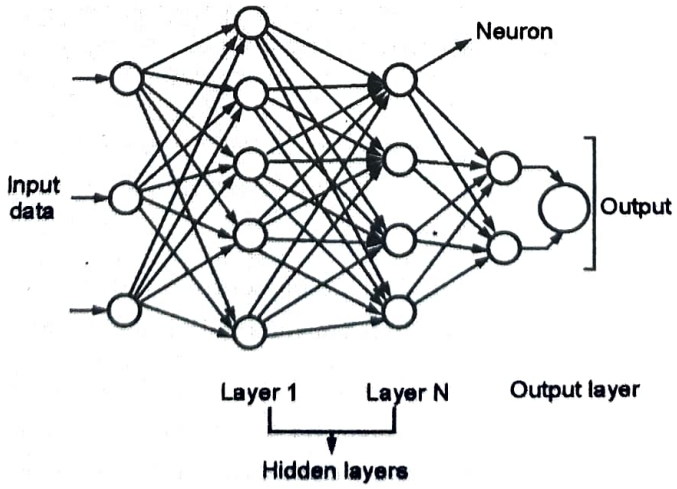


Fig. 6.15.1

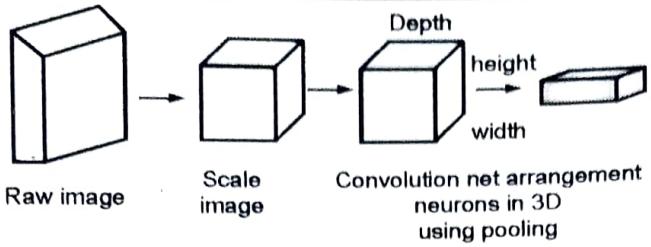


Fig. 6.15.2

In Fig. 6.15.1, there is DNN neural network. On the right, a convolution net arranges it's neurons in 3-dimensions, as seen in one of the layers.

6.15.1 Definition

Convolution is a mathematical operation. In 'convolution neural network' convolution is employed in the network.

6.15.2 Architecture

- A conventional neural network consists of an input layer, hidden layers and output layer. (as shown in the figure). The middle layers are called as hidden layers because their inputs and outputs are governed by activation function and convolution.
- In CNN, the input is a tensor with a shape : (Number of inputs) × (input height) × (input width) × (input channels).
- After passing through a convolutional layer, the image becomes a features map, also called an activation map, with shape :

  (number of inputs) × (feature map height) × (feature map width) × (feature map channels).
- The input is convolved in convolutional layers and the result is forwarded to the next layer. Each convolutional neuron processes data only for it's receptive field.
- Fully connected feed forward neural networks architecture is impractical for larger inputs. It requires a very high number of neurons. For example, a fully connected layer of size 100 × 100 has 10,000 weights for each neuron in the second layer. Instead using convolution a 5 × 5 filing regions, only 25 learnable parameters are required. Also convolutional neural networks are ideal for data with a a grid-like topology since separate features are taken into account during convolution.

6.15.3 Functions of Hidden Layers

GQ. Explain function of Hidden layers.

1. The first hidden layer performs convolution. Each feature map in the hidden layer consists of neurons and each neuron is assigned a receptive field.
2. The second hidden layer performs averaging and subsampling. Each layer consists of feature maps and the neurons of each feature map has a receptive field. It has a trainable bias, trainable coefficient and sigmoid function. They control the operating point of the neuron.

3. The next hidden layer performs a second convolution. Again each feature map in this hidden layer consists of neurons. And each neuron has connections with the previous hidden layer.
4. The next hidden layer performs averaging and subsampling.
5. The output layer performs final stage of convolution. Each neuron is assigned a receptive field and is assigned the possible characters.

The layers in the network alternate between convolution and subsampling, and we get a "bipyramidal" effect. Thus at each layer i.e. either subsampling or convolutional layer, the number of feature maps is increased while the space-resolution is reduced. The weight sharing reduces the number of free parameters in the network compared to the synaptic connections in multilayer perceptron. Also by the use of weight sharing, implementation of convolutional network in parallel form is possible.

6.15.4 Design of Multilayer Perceptron

GQ. Explain in detail Multilayer Perceptron.

- Using convolutional network, a multilayer perceptron of manageable size can learn a complex, high-dimensional, nonlinear mapping.
- Through the training set synaptic weights and bias levels can be learned by simple back propagation algorithm.
- Back propagation algorithm is the key-stone algorithm for the multilayer perceptrons. The partial derivatives of the cost function with respect to the free parameters of the network are determined by back-propagating of the error signals of the output neurons, hence the algorithm is called as Back-Propagation Algorithm.
- Updating the synaptic weights and biases of multilayer perceptron and deriving all partial derivatives of the cost function with respect to free parameters are the base factors for evaluating the power of the algorithm.

Details involved in the **design of a multilayer perceptron** are as follows :

(1) All the neurons in the network are **nonlinear**. And nonlinearity is achieved by using a sigmoid function, and they are
   (i) The nonsymmetric logistic function, and
   (ii) The antisymmetric hyperbolic tangent function.

(2) Each neuron has it's own hyperplane in decision space, and the combination of hyperplanes formed by all the neurons in the network is interatively adjusted by a supervised learning process. And this patterns from different classes are separated with the few classification errors.

(3) For the pattern classification, the random back-propagation algorithm is used to perform the training.

(4) In nonlinear regression, the output range of the multilayer perceptron should be sufficiently larger.

6.15.5 Performance Measure

(i) Algorithm is based on **minimising the cost function**,

(ii) But minimising the cost function may lead to optimising the intermediate quantity, and this may not be the aim of the system.

Hence 'reward-to-volatility' ratio as a performace measure of risk-adjusted return is more appreciable than Eav.

6.15.6 Input Layer

The input layer passes the data directly to the first hidden layer. Here the data is multiplied by the first hidden layers weights. Then the input layer passes the data through the activation function then it passes on.

6.15.7 Pooling Layers

GQ. Explain Pooling layers and its different types.

- Pooling layers are used to reduce the dimensions of the feature maps. It reduces the number of parameters to learn and the amount of computation performed in the network.
- The pooling layer summarises the features present in a region of the feature map generated by a convolutional layer.
- A pooling layers is another building block of a CNN, when processing multichannel input-data, the pooling layer pools each input channel separately.
- Pooling layer reduce the dimensions of the data by combining the output of neuron-clusters. The pooling layer is used to reduce spatial dimensions, but not depth, on a convolutional neural network.
- Pooling layer operates on each feature map independently.
- Pooling is basically 'downscaling' the image obtained from the previous layers. It can be compared to shrinking an image to reduce the pixel density.

6.15.8 Average Pooling

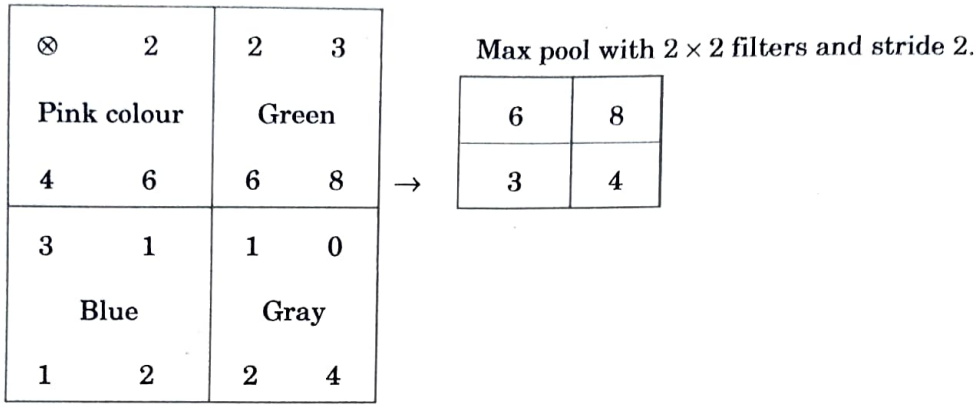
There are two types of widely used pooling in CNN layer.

1. Max pooling	2. Average Pooling

1. **Max-pooling**.

- The popular kind of pooling is **Max-pooling**. Suppose we want to pool by a ratio of 2. It implies that the height and width of your image will be half of it's original value.
- So we have to compress every 4 pixels (a 2×2 grid) and map it to a new single pixel with **loosing** the **important** data from the **missing** pixels.

- Max pooling is done by taking the largest value of these 4 pixels. Thus one new pixel represents 4 old pixels by using the largest value of these 4 pixels. This is repeated for every group of 4 pixels throughout the whole image.



Here the largest pixels are 6,8,3 and 4.

- Here the quality of an image is reduced to avoid computational load on the system. By reducing the quality of the image, we can increase the 'depth' of the layer, to have more features in the reduced image.
- Reducing the image size helps the convolution layer after the pooling layer to look for 'higher level features'. It means that the convolution layer looks at the picture as a whole.

2. Average pooling

- **Average pooling** is different from **Max Pooling**. Average pooling retains 'less important' information about the elements of a block, or pool.
- But Max pooling chooses the maximum value and discards less important values (as shown in the Fig. 6.15.3).

But 'less important' is also sometimes useful in a variety of situations.

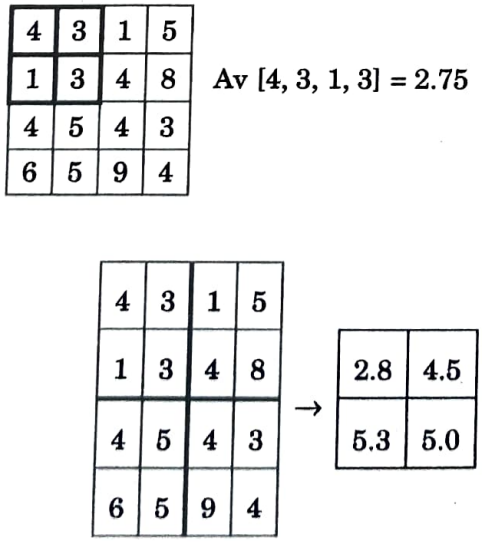


Fig. 6.15.3

6.15.9 Padding

GQ. Discuss padding or Write short note on Padding.

- In Convolutional Neural Networks (CNN), padding is a term that refers to the a number of pixels added to an image when it is being processed by the kernel of a CNN.
- For example, if padding in CNN is kept zero, then every pixel value that is added will be of value zero.
- Padding is simply a process of adding layers of zeros to our input images so as to avoid the problems mentioned above.
- CNNs are used extensively for tackling problems occurring in image processing and predictive modelling or classification tasks. The main application of CNN is to analyse image data.
- Now, every image in any dataset is a matrix of it's pixel values. When working with simple CNN for an image, we get output reduced in size and that is loss of data. And that hinders to obtain a proper result according to our requirements.
- When we do not want the shape or our outputs to reduce in size, the addition of more layers in the data can help to obtain a proper result and that addition can be done by padding.
- Now we study padding with it's importance and how to use it with CNN models. We also see different methods of padding and how they can be implemented.

6.15.10 Problem with Simple Convolution Layer

- A simple CNN of size $(n \times n)$ with $(f \times f)$ filter/kernel size gives result for output image as

  $(n - f + 1) \times (n - f + 1)$
- For example in any convolution operation with a (8×8) image and (3×3) filter the output image size will be (6×6). Thus the output of the layers is shrunk in comparison to the input. Again, the filters we are using may not focus on the corners every time when it moves on the pixels e.g.

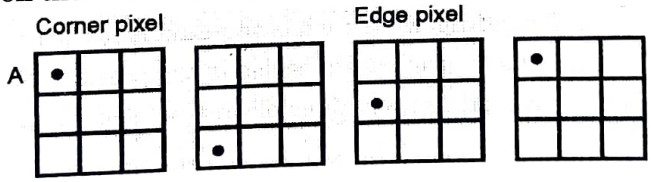


Fig. 6.15.4

- The above image is an example of the movement of a filter of size (3×3) on an image of size (6×6), corner pixel A is coming under the filter in only one movement. This shown that pixel A is misinterpreted.
- This causes loss of information available in the corners and also the output from the layers is reduced and this reduced information may create confusion for next layer. This problem of model can be solved by padding layer.

- The convolution layers reduce the size of the output. So when we want to save the information presented in the corners, we can use padding layers where padding helps by adding extra rows and columns on the outer dimension of the images. So the size of the **input data** will remain similar to the output data.
- Padding basically extends the area of an image in which convolutional neural network processes. The kernel/filter which moves across the image scans each pixels and converts the image into a smaller image.
- Padding is added to the outer frame of the image to allow for more space for the filter to cover in the image. This creates a more accurate analysis of images.

6.15.11 Types of Padding

GQ. What are types of Padding.

We come across three types of padding :

(1) Same padding
(2) Valid Padding
(3) Causal padding

(1) **Same padding :** In this type of padding, the padding layers have zero values in the outer frame of the images or data so the filter we are using can cover the edge of the matrix and it carries the required inference.

(2) **Valid padding :** This type of padding is called as no padding. Here we don't apply any padding, but the input gets, fully covered by the filter and so the every pixel of the image is valid.

Valid padding is to be used with Max-pooling layers. Here we use every pixel or point value while learning the model as valid pixel. It works on the validation of pixel and not on the size of the input.

(3) **Causal padding :** This type of padding works with one-dimensional convolution layers, we can use them majorly in time series analysis. Since a time series is sequential data, it helps in adding zeros at the start of data, and it helps in predicting the values of previous time steps.

We consider an example of a simplified image to understand the idea of edge detection.

Here, a 6×6 matrix convolved with a 3×3 matrix, output is 4×4 matrix. recall that if $n \times n$ image convolved with $f \times f$ kernel or filter then output image is of size.

$$(n - f + 1) \times (n - f + 1)$$

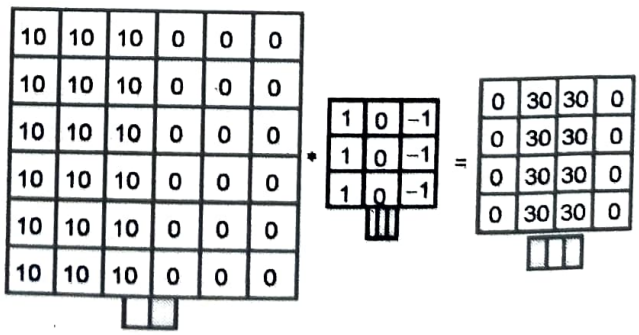


Fig. 6.15.5 : A 6×6 image convolved with 3×3 filter

As we have already seen, there are two problems with convolution :

1. After multiple convolution operation, our original image becomes small which we don't want to happen.
2. Corner features of any image are not used much in the output

Here padding preserves the size of the original image :

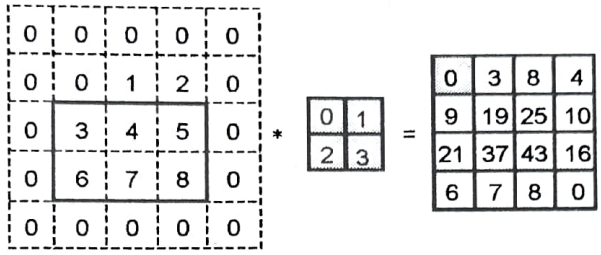


Fig. 6.15.6 : Padded image convolved with 2 * 2 kernel

Hence, if a (n × n) matrix convolved with an (f × f) matrix with padding P then the size of the output image becomes.

$(n + 2P - f + 1) \times (n + 2P - f + 1)$; here $P = 1$

☞ **Remark**

Zero padding is introduced to make the shapes match as required, equally on every side of the input map.

Valid means no padding.

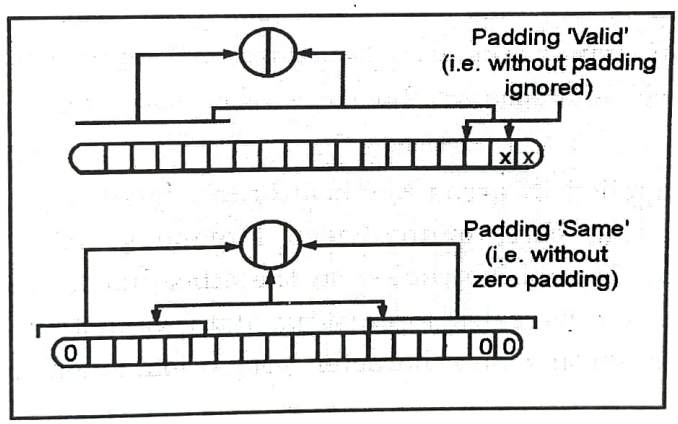


Fig. 6.15.7 : Same versus valid padding with CNN

6.16 MACHINE LEARNING VS NEURAL NETWORK

1. Machine Learning uses advanced algorithms that parse data, learns from it, and use those learnings to discover meaningful patterns of interest. Whereas a Neural Network consists of an assortment of algorithms used in Machine Learning for data modelling using graphs of neurons.
2. While a Machine Learning model makes decisions according to what it has learned from the data, a Neural Network arranges algorithms in a fashion that it can make accurate decisions by itself. Thus,

although Machine Learning models can learn from data, in the initial stages, they may require some human intervention.

Neural networks do not require human intervention as the nested layers within pass the data through hierarchies of various concepts, which eventually makes them capable of learning through their own errors.

3. Machine learning models can be categorized under two types – supervised and unsupervised learning models. However, Neural Networks can be classified into feed-forward, recurrent, convolutional, and modular Neural Networks.

4. An ML model works in a simple fashion – it is fed with data and learns from it. With time, the ML model becomes more mature and trained as it continually learns from the data. On the contrary, the structure of a Neural Network is quite complicated. In it, the data passes through several layers of interconnected nodes, wherein each node classifies the characteristics and information of the previous layer before passing the results on to other nodes in subsequent layers.

5. Since Machine Learning models are adaptive, they are continually evolving by learning through new sample data and experiences. Thus, the models can identify the patterns in the data. Here, data is the only input layer. However, even in a simple Neural Network model, there are multiple layers.

   The first layer is the input layer, followed by a hidden layer, and then finally an output layer. Each layer contains one or more neurons. By increasing the number of hidden layers within a Neural Network model, you can increase its computational and problem-solving abilities.

6. Skills required for Machine Learning include programming, probability and statistics, Big Data and Hadoop, knowledge of ML frameworks, data structures, and algorithms. Neural networks demand skills like data modelling, Mathematics, Linear Algebra and Graph Theory, programming, and probability and statistics.

7. Machine Learning is applied in areas like healthcare, retail, e-commerce (recommendation engines), BFSI, self-driving cars, online video streaming, IoT, and transportation and logistics, to name a few. Neural Networks, on the other hand, are used to solve numerous business challenges, including sales forecasting, data validation, customer research, risk management, speech recognition, and character recognition, among other things.

☞ **Conclusion**

These are some of the major differences between Machine Learning and Neural Networks. Neural Networks are essentially a part of Deep Learning, which in turn is a subset of Machine Learning. So, Neural Networks are nothing but a highly advanced application of Machine Learning that is now finding applications in many fields of interest.

Chapter Ends ...

□□□