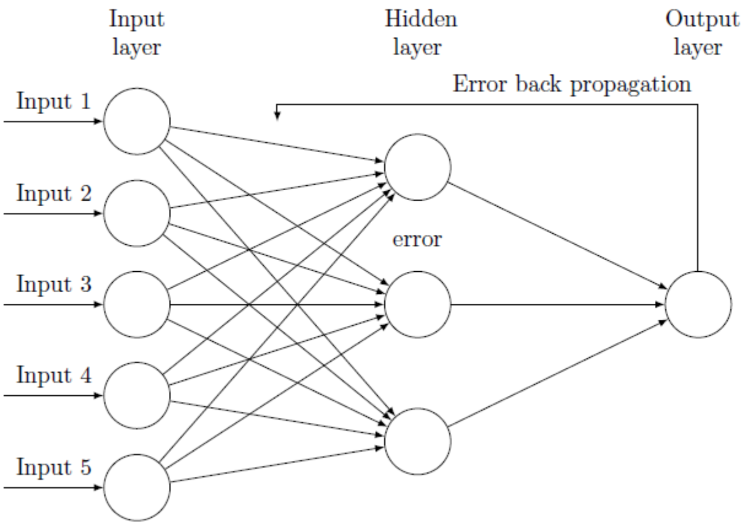
Department of Electronics Engineering

Experiment No. : 05
Error Back Propagation Network

Aim: Write a SCILAB program to implement XOR using error back propagation algorithm.

Apparatus: SCILAB

Circuit Diagram: Error back propagation model:



Theory: Multilayer perceptrons have been applied successfully to solve some difficult and diverse problems by training them in a supervised manner with a highly popular algorithm known as the ***error back-propagation algorithm***. This algorithm is based on the error-correction learning rule. As such, it may be viewed as a generalization of an equally popular adaptive filtering algorithm: the ubiquitous least-mean-square (LMS) algorithm for the special case of a single linear neuron.

Basically, error back-propagation learning consists of two passes through the different layers of the network: a ***forward*** pass and a ***backward*** pass.

- In the forward pass, an activity pattern (input vector) is applied to the sensory nodes of the network, and its effect propagates through the network layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the networks are all fixed.

- During the backward pass, on the other hand, the synaptic weights are all adjusted in accordance with an error-correction rule. Specifically, the actual response of the network is subtracted from a desired (target) response to produce an error signal. This error signal is then propagated backward through the network against the direction of synaptic connections—hence the name "error back-propagation".

- The synaptic weights are adjusted to make the actual response of the network move closer to the desired response in a statistical sense. The learning process performed with the algorithm is called back-propagation learning.

A multilayer perceptron has three distinctive characteristics:

1. The model of each neuron in the network includes a nonlinear activation function. The important point to emphasize here is that the nonlinearity is smooth (i.e., differentiable everywhere), as opposed to the hard-limiting used in Rosenblatt's perceptron. A commonly used form of nonlinearity that satisfies this requirement is a sigmoidal nonlinearity defined by the logistic function: $y_j = \frac{1}{1+\exp(-v_j)}$

   where v_j is the induced local field (i.e., the weighted sum of all synaptic inputs plus the bias) of neuron j, and y_j is the output of the neuron. The presence of nonlinearities is important because otherwise the input-output relation of the network could be reduced to that of a single-layer perceptron. Moreover, the use of the logistic function is biologically motivated, since it attempts to account for the refractory phase of real neurons.

2. The network contains one or more layers of hidden neurons that are not part of the input or output of the network. These hidden neurons enable the network to learn complex tasks by extracting

progressively more meaningful features from the input patterns (vectors).

3. The network exhibits a high degrees of connectivity, determined by the synapses of the network. A change in the connectivity of the network requires a change in the population of synaptic connections or their weights.

Algorithm : Initialize the weights and learning rate to some small random values. Repeat the steps when the stopping condition is false.

PHASE – I : Feedforward phase

1. Each input unit receives input signal x_i and sends it to the hidden unit (i = 1 to n)
2. Each hidden unit z_j (j = 1 to p) sums its weighted input signals to calculate net input.

$$z_{in\,j} = v_{0\,j} + \sum_{i=1}^{n} x_i v_{i\,j}$$

3. Apply the activation function, $z_j = f(z_{in\,j})$
4. For each output unit, calculate net input. $y_{ink} = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$

and apply the activation function to compute the output signal.

$$y_k = f(y_{ink})$$

PHASE – II : Back propagation of error

5. Each output unit receives a target pattern corresponding to the input training pattern and computes the error correction term :

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

on the basis of the calculated error correction term, update the change in weights and bias, $\Delta w_{jk} = \alpha \delta_k z_j$

$$\Delta w_{0k} = \alpha \delta_k$$

send the correction to the hidden layers.

6. Each hidden unit z_j, sums its delta input from the output units.

$$\delta_{in\,j} = \sum_{k=1}^{m} \delta_k w_{jk}$$

$$\delta_j = \delta_{in\,j} f'(z_{in\,j})$$

The derivative of function can be calculated depending on whether binary or bipolar sigmoidal function is used. On the basis of the calculated δ_j, update the change in weights and bias as,

$$\Delta v_{ij} = \alpha \delta_j x_i$$
$$\Delta v_{0j} = \alpha \delta_j$$

PHASE – III : Weight and Bias updation

7. Each output unit updates the weights and bias as,

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$
$$w_{0k}(new) = w_{0k}(old) + \Delta w_{0k}$$

Each hidden unit updates the weights and bias as,

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$
$$v_{0j}(new) = v_{0j}(old) + \Delta v_{0j}$$

8. Check for the stopping condition.

The stopping condition may be a certain number of epochs reached or when the actual output equals the target output.

Problem :
Using back-propagation network, find the new weights for the net. It is presented with the input pattern [0.6 0.8 0] and the target output is 0.9.
Use a learning rate α = 0.3 and binary sigmoid activation function.
The initial weights are given as,

$[w_{11}\ w_{21}\ w_{31}\ b_1] = [2\ 1\ 1\ -1]$

$[w_{12}\ w_{22}\ w_{32}\ b_2] = [1\ 2\ 3\ 1]$

$[w_{13}\ w_{23}\ w_{33}\ b_3] = [0\ 2\ 1\ -1]$

$[v_{11}\ v_{21}\ v_{31}\ b_3] = [-1\ 1\ 2\ -1]$

Results :

1. **Solve the problem step by step and derive the value of weights after first iteration. Attach the images for the solved problem in the space below.**

Experiment 5
EBPN

60001180046
Reeha Parkar
24/09/2021
1 of 4

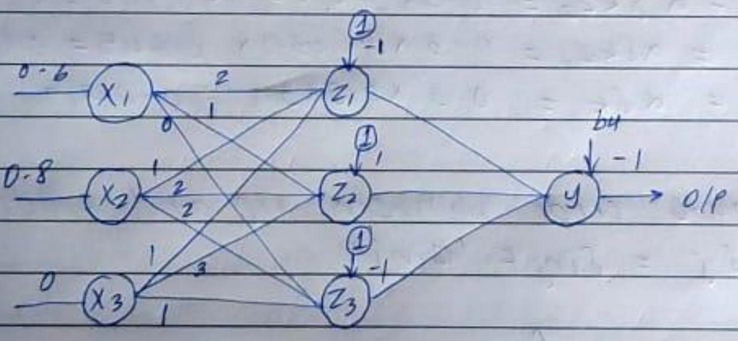
Input pattern given = [0.6 0.8 0]
$\alpha = 0.3$ [learning rate]
Target output = 0.9

$[w_{11}\ w_{21}\ w_{31}\ b_1] = [2\ 1\ 1\ -1]$
$[w_{12}\ w_{21}\ w_{32}\ b_2] = [1\ 2\ 3\ 1]$
$[w_{13}\ w_{23}\ w_{33}\ b_3] = [-1\ 1\ 2\ -1]$

Binary sigmoid activation function:

$$f(x) = \frac{1}{1+e^{-ax}} \quad ; a = 1$$

Neural Network:



Step I: Net input to hidden neurons:

$z_{in}1 = 0.6 \times 2 + 0.8 \times 1 + (-1) = 1$; $z_1 = f(z_{in}1) = 0.731$
$z_{in}2 = 0.6 \times 1 + 0.8 \times 2 + 1 = 3.2$; $z_2 = f(z_{in}2) = 0.960$
$z_{in}3 = 0.8 \times 2 - 1 = 0.6$; $z_3 = f(z_{in}3) = 0.643$

Step II: Calculating net input to O/P neuron:

$y_{in} = -1(0.731) + 1(0.960) + 2(0.645)$
$y_{in} = 0.519$
$y = f(y_{in}) = 0.6269$

60001180046

24/09/2021
2 of 4

Step III: Error portion calculation:

$\delta_k = (t_k - y_k)\, f'(y_{in\,k})$

$f'(y_{in\,k}) = \lambda y(1-y)$

$= 1 \times 0.6269 (1 - 0.6269)$

$f'(y_{in}) = 0.2338$

$\delta_k = (0.9 - 0.6269)\, 0.2338$

$\delta_k = 0.0639$

Step IV: change of weights from hidden neuron to o/p neuron:

$\Delta V_{11} = \alpha \delta_k z_1 = 0.3 \times 0.0639 \times 0.731 = 0.0140$

$\Delta V_{22} = \alpha \delta_k z_2 = 0.3 \times 0.0639 \times 0.960 = 0.0184$

$\Delta V_{31} = \alpha \delta_k z_3 = 0.3 \times 0.0639 \times 0.645 = 0.0123$

$\Delta b_4 = \alpha \delta_k = 0.3 \times 0.0639 = 0.01917$

Step V: compute error between i/p and hidden layer.

$\delta_j = \delta_{inj}\, f'(z_{inj})$

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

$\delta_{in1} = \delta_k v_{jk} = \delta_k V_{11} = -0.0639$

$\delta_{in2} = \delta_k V_{21} = 0.0639$

$\delta_{in3} = \delta_k V_{31} = 0.1278$

$f'(z_{in1}) = z_1(1-z_1) = 0.1986$

$f'(z_{in2}) = z_2(1-z_2) = 0.0384$

$f'(z_{in3}) = z_3(1-z_3) = 0.2289$

60001180046

24/09/2021
3 of 4

$\delta_1 = \delta in_1 \; f'(zin_1) = -0.639 \times 0.1986 = -0.01$
$\delta_2 = \delta in_2 \; F'(zin_2) = 0.0024$
$\delta_3 = \delta in_3 \; f'(zin_3) = 0.00293$

Step VI: weight changes between i/p and hidden layer.

$\Delta w_{11} = \alpha \delta_1 x_1 = 0.3(-0.0125) \times 0.6 = -0.0022.$
$\Delta w_{21} = \alpha \delta_1 x_2 = 0.3(-0.0125) \times 0.8 = -0.0030$
$\Delta w_{31} = \alpha \delta_1 x_3 = 0.3(-0.0125) \times 0 = 0$
$\Delta w_{12} = \alpha \delta_2 x_1 = 0.3(0.0024) \times 0.6 = 0.00043$
$\Delta w_{22} = \alpha \delta_2 x_2 = 0.3(0.0024) \times 0.8 = 0.00057$
$\Delta w_{32} = \alpha \delta_2 x_3 = 0.3(0.0024) \times 0 = 0$
$\Delta w_{13} = \alpha \delta_3 x_1 \; 0.3(0.00293) \times 0.6 = 0.0052$
$\Delta w_{23} = \alpha \delta_3 x_2 = 0.3(0.00293) \times 0.8 = 0.0069.$
$\Delta w_{33} = \alpha \delta_3 x_3 = 0.3(0.00293) \times 0 = 0$

$\Delta b_1 = \alpha \delta_1 = 0.3 \times (-0.00125) = -0.0037$
$\Delta b_2 = \alpha \delta_2 = 0.3 \times (0.0024) = 0.0007$
$\Delta b_3 = \alpha \delta_3 = 0.3 \times (0.00293) = 0.0088.$

Step VII: Update weights:

$w_{11}(new) = w_{11}(old) + \Delta w_{11} = 2 + (-0.0022) = 1.997$
$w_{21}(new) = 1 + (-0.003) = 0.9969.$
$w_{31}(new) = 1 + 0 = 1$
$w_{12}(new) = -1 - 0.0037 = -1.0037$
$w_{22}(new) = 2 + 0.005 = 2.0005$
$w_{32}(new) = 3 + 0 = 3$
$w_{13}(new) = 0 + 0.0052 = 0.0052$
$w_{23}(new) = 2 + 0.0069 = 2.0069$
$w_{33}(new) = 1 + 0 = 1$

60001180046

24/09/2021
4 of 4

$b_1(new) = -1 - 0.0037 = -1.0037$

$b_2(new) = 1 + 0.0007 = 1.0007$

$b_3(new) = -1 + 0.0087 = -0.9913$

2. Code and output

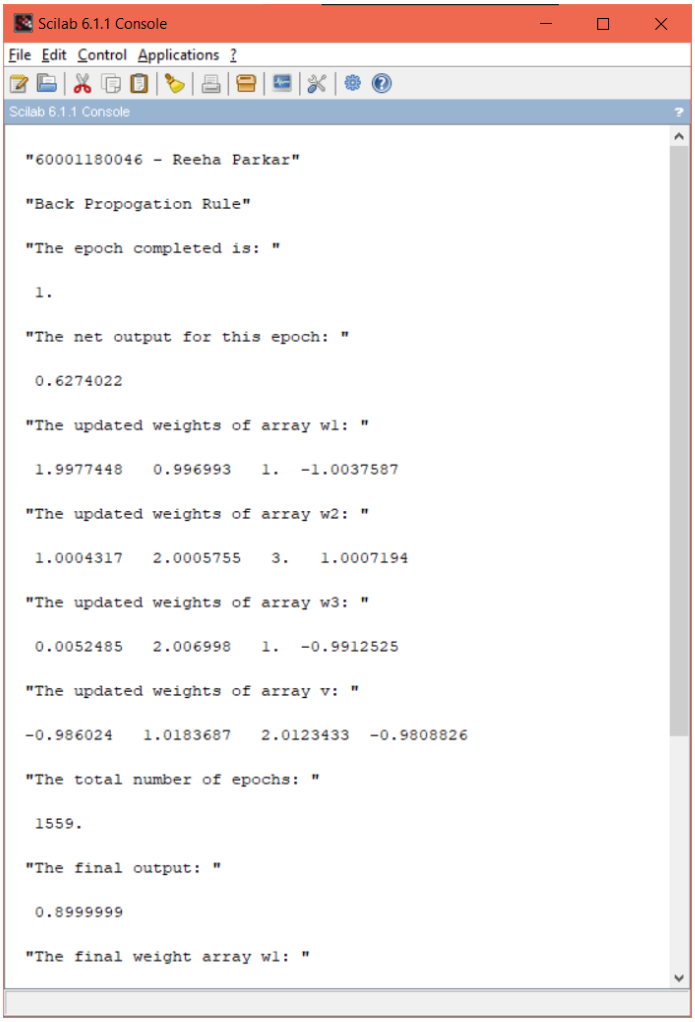
Code:

```
clc;
clear;

disp("60001180046 - Reeha Parkar");
disp("Back Propogation Rule");

//Input Pattern
x = [0.6 0.8 0];

//Bias
b = [1 1 1 1];

//Learning Rate
L = 0.3;

//Initial Weights
w1 = [2 1 1 -1];
w2 = [1 2 3 1];
w3 = [0 2 1 -1];
v = [-1 1 2 -1];

//Target
t = 0.9;
epoch = 0;
y = 0;

//Training Algorithm
while t - y >= 0.0000001

    //General Declaration
    zin1 = 0;
    zin2 = 0;
    zin3 = 0;
    z = [];
    yin = 0;
    y = 0;

    //Net output to hidden neuron
    epoch = epoch + 1;

    for i = 1:3
        zin1 = x(i)*w1(i) + zin1;
        zin2 = x(i)*w2(i) + zin2;
        zin3 = x(i)*w3(i) + zin3;
    end
    zin1 = b(1)*w1(4) + zin1;
    zin2 = b(2)*w2(4) + zin2;
    zin3 = b(3)*w3(4) + zin3;
    zin = [zin1 zin2 zin3];
    //disp(zin);

    for i = 1:3
        z(i) = 1/(1 + exp(-zin(i)));
    end
    //disp(z);

   //Calculate net input to output neuron
    for i = 1:3
        yin = z(i)*v(i) + yin;
```

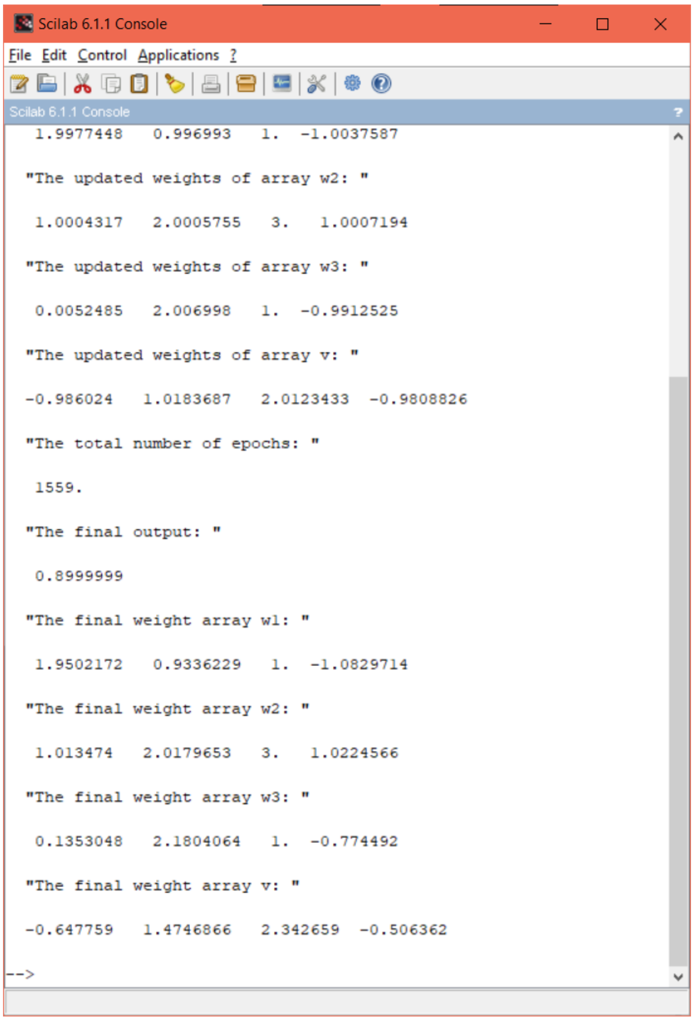
```
end

yin = b(4)*v(4) + yin;
y = 1/(1 + exp(-yin));

//Compute the error "Dk"
f = 1*y*(1-y);
Dk = (t - y)*f;

//Finding the Change of weights between hidden & output layer
for i = 1:3
    Dv(i) = L*Dk*z(i)
end
Dbv = L*Dk;
//disp(Dv);

//Compute the error portion between input and hidden layer "Dj"
for i = 1:3
    Din(i) = Dk*v(i);
end

for i = 1:3
    f(i) = 1*z(i)*(1-z(i));
end

for i = 1:3
    D(i) = Din(i)*f(i);
end

//Weight change between input & hidden layer
for i = 1:3
    Dw1(i) = L*D(1)*x(i);
    Dw2(i) = L*D(2)*x(i);
    Dw3(i) = L*D(3)*x(i);
end

for i = 1:3
    Db(i) = L*D(i);
end

//Updating the weights
for i = 1:3
    w1(i) = w1(i) + Dw1(i);
    w2(i) = w2(i) + Dw2(i);
    w3(i) = w3(i) + Dw3(i);
    v(i) = v(i) + Dv(i);
end

w1(4) = w1(4) + Db(1);
w2(4) = w2(4) + Db(2);
w3(4) = w3(4) + Db(3);
v(4) = v(4) + Dbv;

if epoch==1
    disp("The epoch completed is: ");
    disp(epoch);
    disp("The net output for this epoch: ");
    disp(y);
    disp("The updated weights of array w1: ");
    disp(w1);
    disp("The updated weights of array w2: ");
    disp(w2);
    disp("The updated weights of array w3: ");
    disp(w3);
    disp("The updated weights of array v: ");
```

```
        disp(v);
    end
end

disp("The total number of epochs: ");
disp(epoch);
disp("The final output: ");
disp(y);
disp("The final weight array w1: ");
disp(w1);
disp("The final weight array w2: ");
disp(w2);
disp("The final weight array w3: ");
disp(w3);
disp("The final weight array v: ");
disp(v);
```

Output:



```
Scilab 6.1.1 Console
File Edit Control Applications ?
Scilab 6.1.1 Console

 "60001180046 - Reeha Parkar"

 "Back Propogation Rule"

 "The epoch completed is: "

  1.

 "The net output for this epoch: "

  0.6274022

 "The updated weights of array w1: "

  1.9977448   0.996993   1.   -1.0037587

 "The updated weights of array w2: "

  1.0004317   2.0005755   3.   1.0007194

 "The updated weights of array w3: "

  0.0052485   2.006998   1.   -0.9912525

 "The updated weights of array v: "

 -0.986024   1.0183687   2.0123433   -0.9808826

 "The total number of epochs: "

  1559.

 "The final output: "

  0.8999999

 "The final weight array w1: "
```



```
Scilab 6.1.1 Console
File Edit Control Applications ?
Scilab 6.1.1 Console
  1.9977448   0.996993   1.   -1.0037587

 "The updated weights of array w2: "

  1.0004317   2.0005755   3.   1.0007194

 "The updated weights of array w3: "

  0.0052485   2.006998   1.   -0.9912525

 "The updated weights of array v: "

 -0.986024   1.0183687   2.0123433   -0.9808826

 "The total number of epochs: "

  1559.

 "The final output: "

  0.8999999

 "The final weight array w1: "

  1.9502172   0.9336229   1.   -1.0829714

 "The final weight array w2: "

  1.013474   2.0179653   3.   1.0224566

 "The final weight array w3: "

  0.1353048   2.1804064   1.   -0.774492

 "The final weight array v: "

 -0.647759   1.4746866   2.342659   -0.506362

-->
```

Conclusion :
In this experiment, we wrote a SCILAB program to implement XOR using error back propagation algorithm. The target output that is 0.9, was successfully obtained experimentally to the value 0.8999999. This proves that our coded EBPN network is accurate.

Review Questions

Answer the following questions on journal sheets and attach the images or scan a pdf for the same.

1. What is meant by epoch in a training process?
2. List the stages involved in the training of the error back propagation algorithm.
3. State the significance of error portions of each stage in the error back propagation algorithm.
4. What activation functions can be used in the error back propagation algorithm?
5. State and explain the concept of generalization with respect to the error back propagation algorithm.

Reeha Parkar
60001180046

NNFL Review Questions
Experiment 5: EBPN.

Q1. What is meant by epoch in a training process?

- An epoch is a term used in machine learning and indicates the number of passes of the entire training set the machine learning algorithm has completed.
- Determining how many epochs a model should run to train is based on many parameters related to both the data itself and the goal of the model.

Q2. List the stages involved in the training of the error back propagation algorithm.

- Initialize the weights and ~~that~~ learning rate to some small random values.
- Repeat the steps till when the stopping condition is false.

Phase I : Feedforward phase.

1. Each input unit receives input signal x_i and sends it to the hidden unit $(i=1$ to $n)$
2. Each hidden unit z_j $(j=1$ to $p)$ sums its weighted input signals to calculate the net input
$$z_{inj} = v_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$
3. Apply the activation function.
4. For each output unit, calculate net input.
$$y_{ink} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk}$$
and apply activation function to compute the output signal. $y_k = f(y_{ink})$

FOR EDUCATIONAL USE

60001180046.

2 of 4.

Phase II : Back propagation of error.

5. Each output unit receives a target pattern corresponding to the input training pattern and computes the error correction term.

$$\delta_k = (t_k - y_k)\, f'(y_{in\,k})$$

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

$$\Delta w_{jk} = \alpha\, \delta_k z_j$$
$$\Delta w_{0k} = \alpha\, \delta_k.$$

6. Each hidden unit z_j, sums its delta input from the output units

$$\delta_{in\,j} = \sum_{k=1}^{m} \delta_k\, w_{jk}$$
$$\delta_j = \delta_{inj}\, f'(z_{inj})$$

The derivative of the function can be calculated depending on whether binary or bipolar sigmoidal function is used.

On the basis of the calculated δ_j, update the change in weights and bias as,

$$\Delta v_{ij} = \alpha\, \delta_j x_i$$
$$\Delta v_{0j} = \alpha\, \delta_j$$

Phase III: Weight and Bias updation:

7. Each output unit updates the weights and bias as:

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$
$$w_{0k}(new) = w_{0k}(old) + \Delta w_{0k}$$

Each hidden unit updates the weights and bias as:

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$
$$v_{0j}(new) = v_{0j}(old) + \Delta v_{0j}$$

8. check for stopping condition. which can be a certain no. of epochs or when target = actual output.

Sundaram

FOR EDUCATIONAL USE

60001180046.

3 of 4

Q3. What activation functions can be used in the error back propagation algorithm?

Binary sigmoidal activation:

$f'(y_{in}) = \lambda f(y_{in})[1 - f(y_{in})]$

$\therefore y = \lambda y [1 - y]$

Bipolar sigmoidal activation:

$f'(y_{in}) = \frac{\lambda}{2} [1 - f(y_{in})][1 + f(y_{in})]$

$\therefore y' = \frac{\lambda}{2} [1 - y][1 + y]$

Q4. State the significance of error portions of each stage in the error back propagation algorithm.

- Backpropagation follows the concept of backward propagation of error, uses gradient descent.
- Each output unit receives a target pattern corresponding to the input training pattern and computes the error correction term (δ_k).
- The calculation of the gradient proceeds backwards through the network. Partial computations of the first or one layer are reused in the computation of gradient for the previous layer.
- This backward flow of error allows efficient computation of the gradient for the previous layer.
- On the basis of the calculated error correction term, the weights and bias are updated.

FOR EDUCATIONAL USE

60001180046.

4 of 4

Q5. State and explain the concept of generalization with respect to error backpropagation algorithm.

The backpropagation algorithm is applied to multilayer feed forward networks consisting of continuous differentiable activation function.
For a given set of training I/O pairs, the algorithm provides a procedure for changing the weights in a BPN to classify the given input patterns correctly.
The aim of the neural network is to train the net to achieve a balance the net's ability to respond (memorization) and its ability to give reasonable responses to the input that is similar but not identical to the one that is used in training (generalization).
NN generalizes well, if the I/O mapping computed by the network is nearly correct for new data.
Factors that influence generalization:
- size of the training set
- architechture of NN
- complexity of problem at hand.