Experiment 4: Back propogation for Exclusive OR gate

Table of Contents

Name: Ventrapragada Sai Shravani

PRN:17070123120

Batch:Entc(2017-21) G-3

Aim:

Implementation of EX-OR gate using back propagation

Theory:

Back propagation comes under a multiple layer feed forward networks.Back propogation is a supervised learning network, it consists of input, hidden and output layers.

A feedback is provided to the hidden and output layer and input layer does not contain bias since there is no feedback present.

This algorithm can be explained in 2 phases: Feed-forward phase, back-propogation phase of learning where signals are sent in reverse direction.

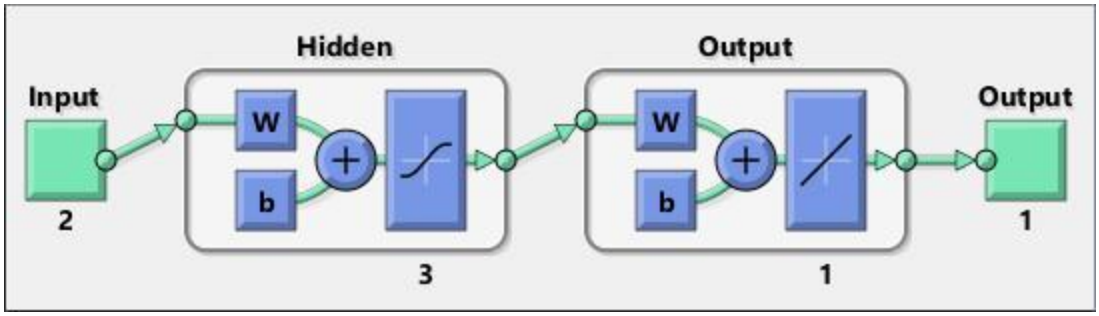
Gradient descent: Is an optimisation algorithm used to find the value of parameter/ coefficient of a function, that minimises a cost function.

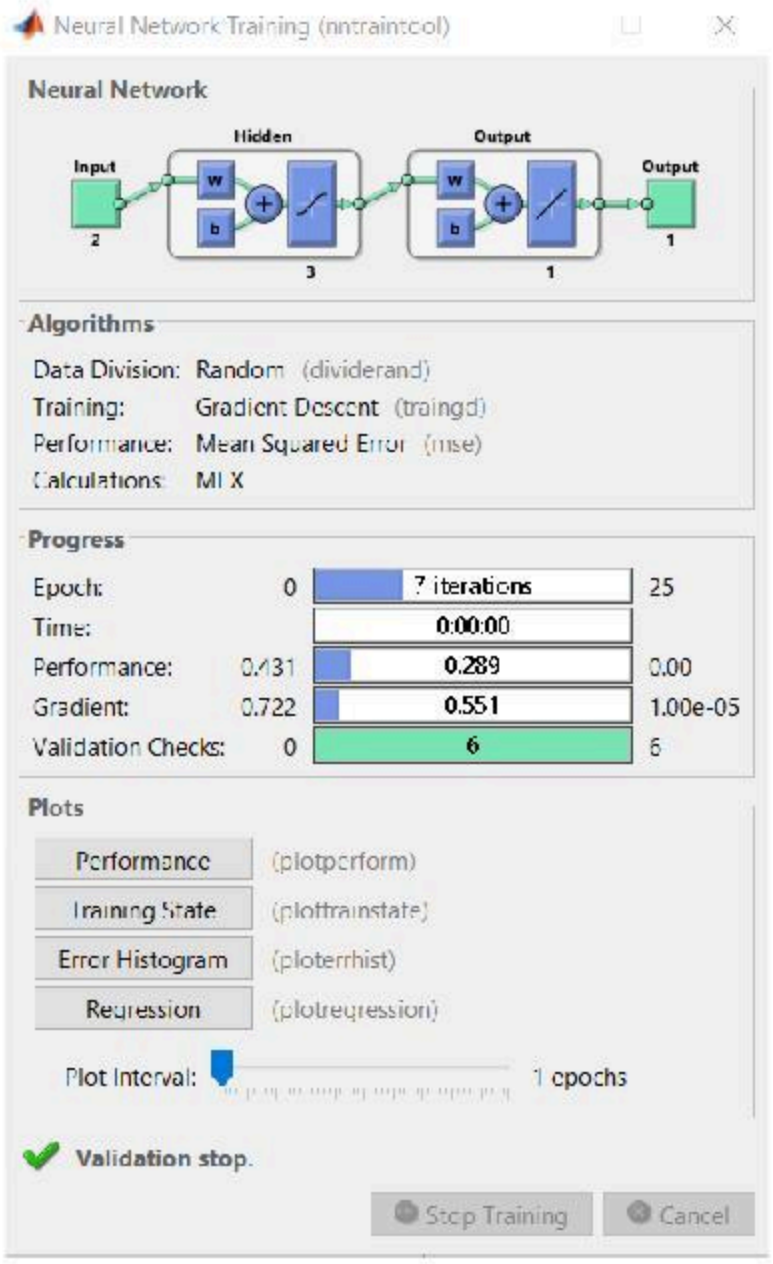
Back Propogation for EX-OR gate

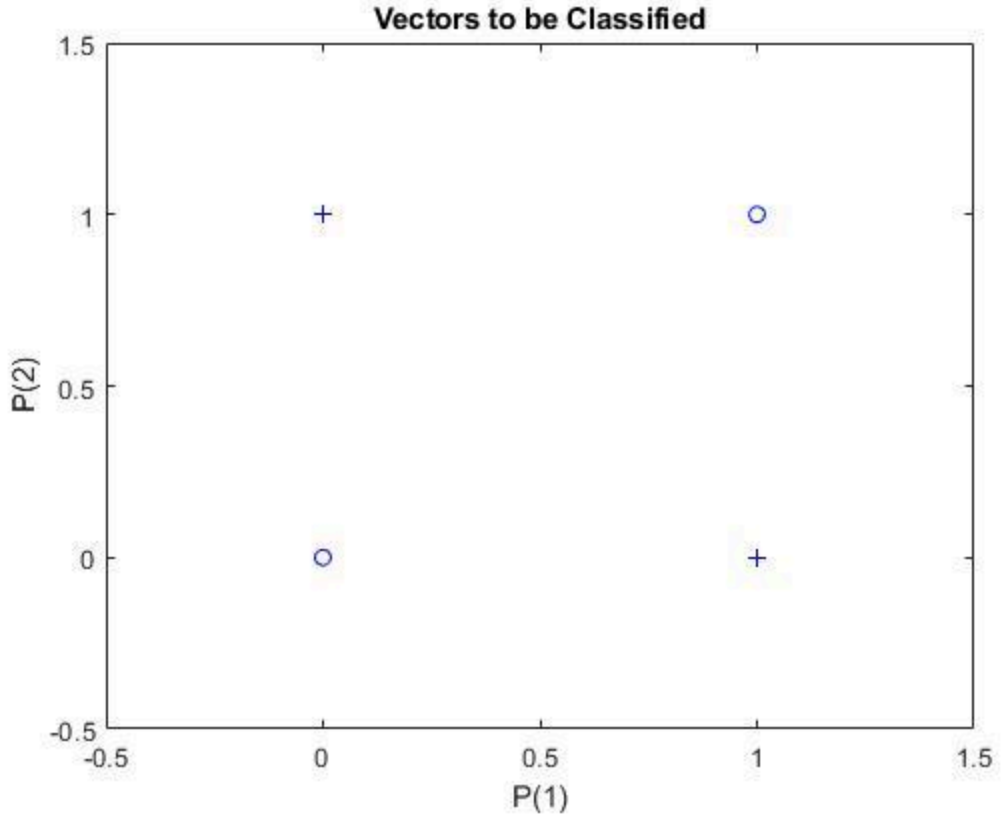
```
clc;
clear all;
close all;
X = [[1;0],[0;1], [0;0],[1;1]];  % This defines 4 input training
 vectors
t = [1 1 0 0];    % This defines the corresponding output for each
 vector
net = feedforwardnet(3,'traingd'); % 3 defines the input out and
 hidden layer
% Modifying the defined parameters
```

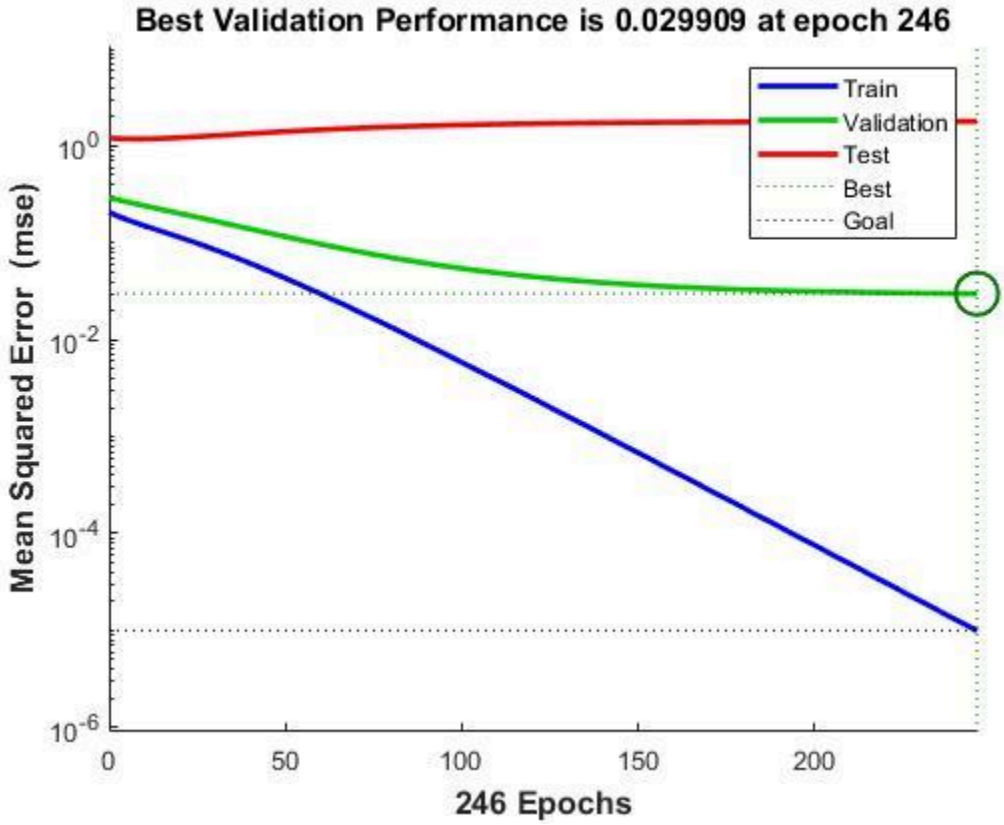
```
net.trainParam.show =50;
net.trainParam.lr = 0.05;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
%The goal of "training" a neural network is to find the right weights
 that
%correctly predict what the output of an given input is. There are two
 fundamental
%processes that go into training a neural network.
[net,tr] = train(net,X,t); %Training the network
%tr will contain all the details about the training
view(net)
y = net(X)
sim(net, [1; 1]) % test the network with the input [0;0]
figure(1)
figure(2)
i=imread('C:\Users\acer\Documents\4th year\Final year Project
\capture1.png');
imshow(i);
figure(3)
plotpv(X, t)     % plot the network training data
figure(4)
plotperform(tr)
figure(5)
plottrainstate(tr)
```

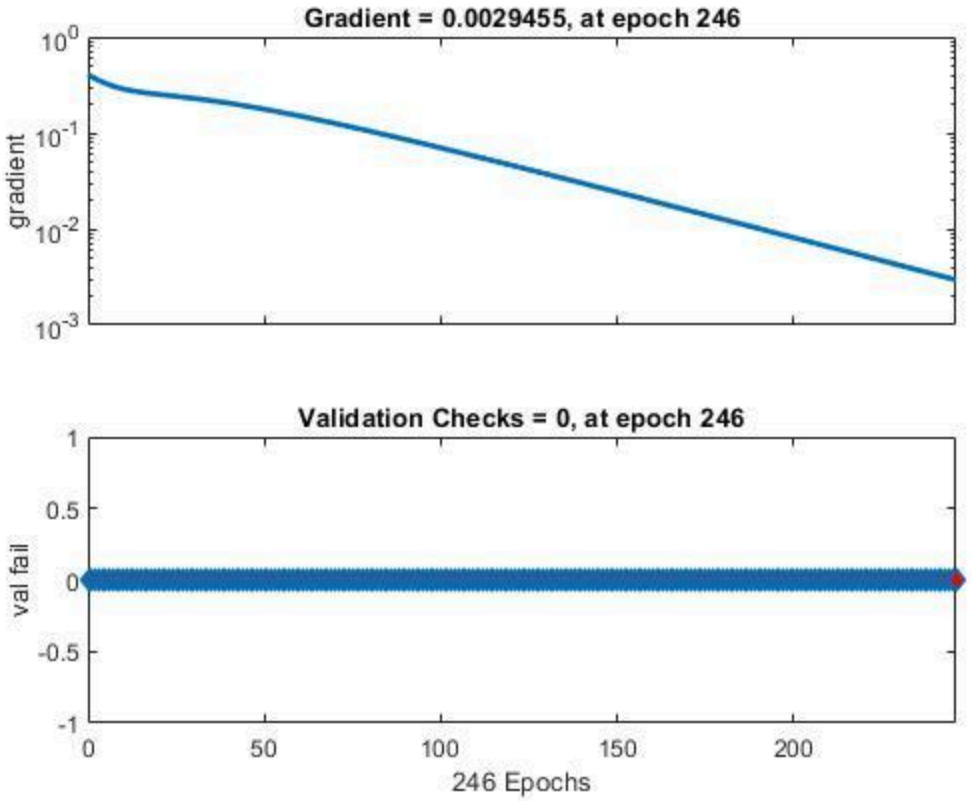
```
y =

    0.8271    0.9970    0.0033    1.3389


ans =

    1.3389
```




Neural Network Training (nntraintool)
Neural Network
Input
Hidden
Output
Output
W
b
2
3
1
1
Algorithms
Data Division: Random (dividerand)
Training: Gradient Descent (traingd)
Performance: Mean Squared Error (mse)
Calculations: MEX
Progress
Epoch: 0 7 iterations 25
Time: 0:00:00
Performance: 0.431 0.289 0.00
Gradient: 0.722 0.551 1.00e-05
Validation Checks: 0 6 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Plot Interval: 1 epochs
Validation stop.
Stop Training
Cancel


Vectors to be Classified
1.5
1
0.5
0
-0.5
P(2)
-0.5
0
0.5
1
1.5
P(1)


Best Validation Performance is 0.029909 at epoch 246
Train
Validation
Test
Best
Goal
Mean Squared Error (mse)
10^{0}
10^{-2}
10^{-4}
10^{-6}
0
50
100
150
200
246 Epochs



Conclusion

In this code we observed graphs of performance, calculated and updated biases, also found final weights after upgradation. Also ploted gradient descent graph in figure 5. And simulated a test data and obtained desired output.

Published with MATLAB® R2020a