Assignment 5

Experiment 11

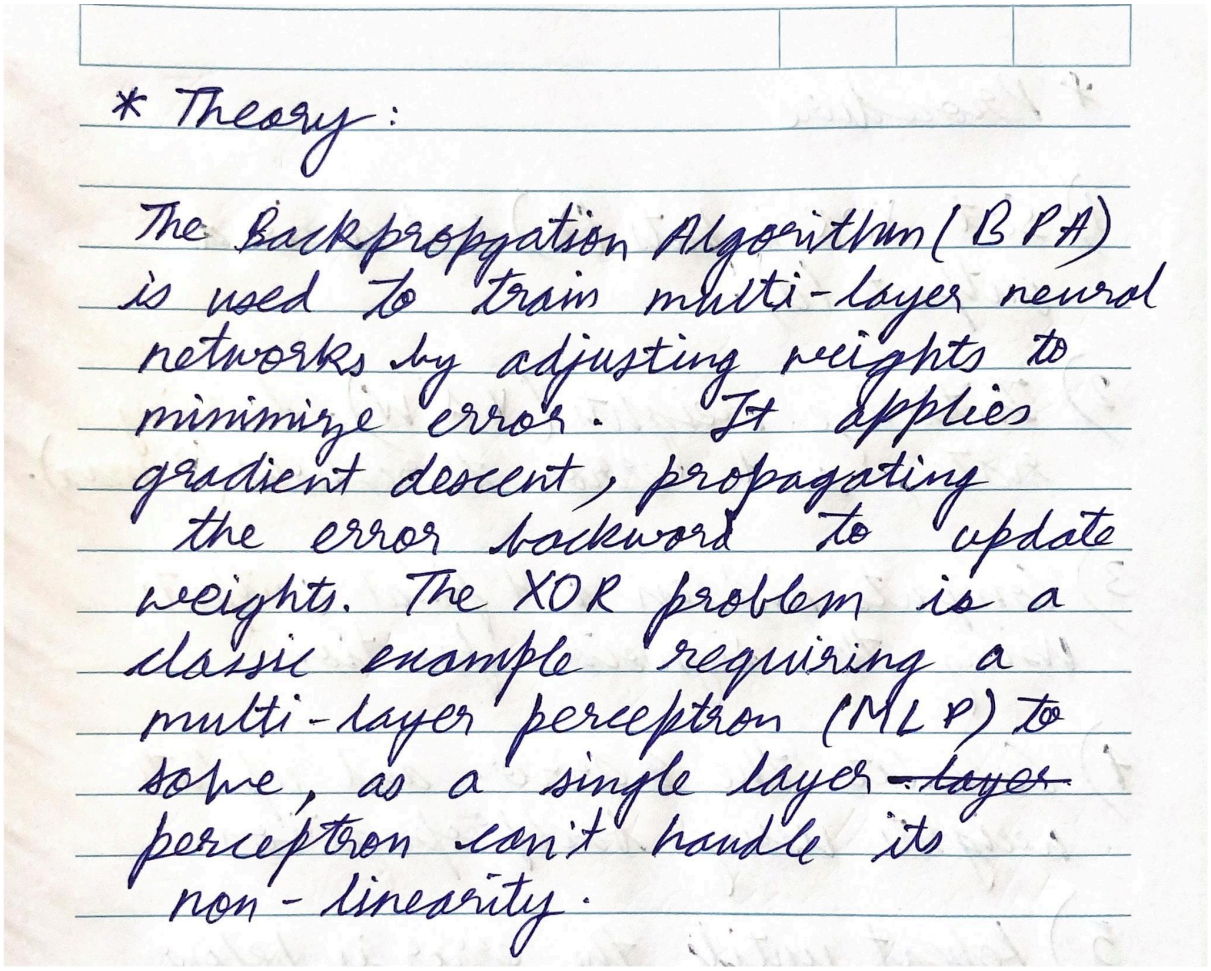
Title: Implement BPA to train XOR problem

Name of Student: Sangeet Agrawal PRN No. 21070122140

DoP: 7 Oct DoS: 9 Oct

Aim: To design and simulate a Backpropagation training algorithm for multi-layer continuous perception.

Problem Statement: Implement BPA to train XOR problem



* Theory:

The Backpropagation Algorithm (BPA) is used to train multi-layer neural networks by adjusting weights to minimize error. It applies gradient descent, propagating the error backword to update weights. The XOR problem is a classic example requiring a multi-layer perceptron (MLP) to solve, as a single layer ~~layer~~ perceptron can't handle its non-linearity.

* Procedure

1) Set XOR inputs (X) and desired outputs (d).

2) Initialize weights (V & W), learning rate, and error threshold (Emax).

3) Compute hidden and final outputs using the sigmoid function.

4) Calculate the error and update weights using backpropagation.

5) Repeat until the error is below Emax or max iterations are reached.

Inputs:

```
X = [0 0; 0 1; 1 0; 1 1];
d = [0; 1; 1; 0];
V = [0.5 0.5; 1 0.3];
W = [0.5 0.5];
```

Code:

```
X = [0 0; 0 1; 1 0; 1 1];
d = [0; 1; 1; 0];
V = [0.5 0.5; 1 0.3];
W = [0.5 0.5];

Emax = 0.1;
learning_rate = 1;
max_iters = 10000;

% Activation function and its derivative
f = @(x) 1 ./ (1 + exp(-x));
f_prime = @(x) f(x) .* (1 - f(x));

for iter = 1:max_iters
   total_error = 0;

   for p = 1:size(X, 1)
       x = X(p, :)';
       y = f(V * x);
       o = f(W * y);

       dk = d(p);
       ok = o;
       e = 0.5 * (dk - ok)^2;
       total_error = total_error + e;


       dO = (dk - ok) * f_prime(ok);
       dY = (W' * dO) .* f_prime(y);

       % Update weights
       W = W + learning_rate * dO * y';  % Update weights from
hidden to output layer
       V = V + learning_rate * dY * x';
   end

   if total_error < Emax
       fprintf('Training converged at iteration %d with total
error %f\n', iter, total_error);
       break;
```

```
    end
end
disp('Final weights (V from input to hidden):');
disp(V);
disp('Final weights (W from hidden to output):');
disp(W);
```

Output:

```
Command Window
>> Experiment_11
Final weights (V from input to hidden):
   -4.0847   -5.1276
   -3.6710   -4.7456

Final weights (W from hidden to output):
  -20.9374   13.7595

>>
```