MATLAB Deep Learning Notes IV

Key Words : Comparison among SGD, Batch and Mini Batch; Limitations of Single-Layer Neural Networks

1. Comparison among SGD, Batch and Mini Batch

1. We investigate the learning speeds of the SGD, Batch and Mini Batch. The errors of these schemes are compared at the end of the training processes for the entire training data. It is worth pointing out that the weights of these three methods are initialized with the same values in order to evaluate a fair comparison.

Program 8: SGDvsBatchvsMiniBatch

Listing 1: SGDvsBatchvsMiniBatch.m

```
clear

data_input = [ 0, 0, 1; 0, 1, 1; 1, 0, 1; 1, 1, 1 ]; % training data
correct_output = [ 0; 0; 1; 1 ]; % correct outputs(i.e., labels)
weight1 = 2 * rand(1, 3) - 1; % initializes the weights with random real
    numbers between [-1, 1]
weight2 = weight1;
weight3 = weight1;

E1 = zeros(1000, 1);
E2 = E1;
E3 = E1;



for epoch = 1 : 1000 % the number of epoch = 1000
    weight1 =  DeltaSGD(weight1, data_input, correct_output);
    weight2 =  DeltaBatch(weight2, data_input, correct_output);
    weight3 =  DeltaMiniBatch(weight3, data_input, correct_output);

    es1 = 0; es2 = 0; es3 = 0;
    N = 4;
```

```
    for k = 1 : N
        x = data_input(k, :)';
        d = correct_output(k);
        % SGD
        v1 = weight1 * x;
        y1 = sigmoid(v1);
        es1 = es1 + (d - y1)^2;
        % Batch
        v2 = weight2 * x;
        y2 = sigmoid(v2);
        es2 = es2 + (d - y2)^2;
        % Mini Batch
        v3 = weight3 * x;
        y3 = sigmoid(v3);
        es3 = es3 + (d - y3)^2;
    end
    E1(epoch) = es1 / N;
    E2(epoch) = es2 / N;
    E3(epoch) = es3 / N;

end

plot(E1, '--b', 'Linewidth',1);
hold on
plot(E2, '-.g', 'Linewidth',1);
hold on
plot(E3, '-r', 'Linewidth',1);
xlabel('the number of Epoch')
ylabel('Average of Training error')
legend('SGD', 'Batch', 'Mini Batch')
```

Output 8:

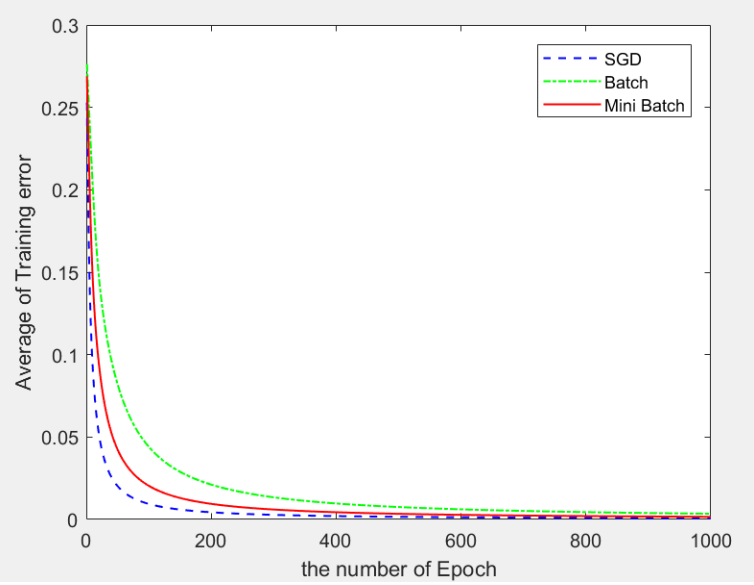

Figure 1: The SGD method learns faster than the Batch method and Mini Batch method

2. Limitations of Single-Layer Neural Networks

2. Consider the same neural network that was discussed in the previous section. It consists of three input nodes and an output node, and the activation function of the output node is a sigmoid function

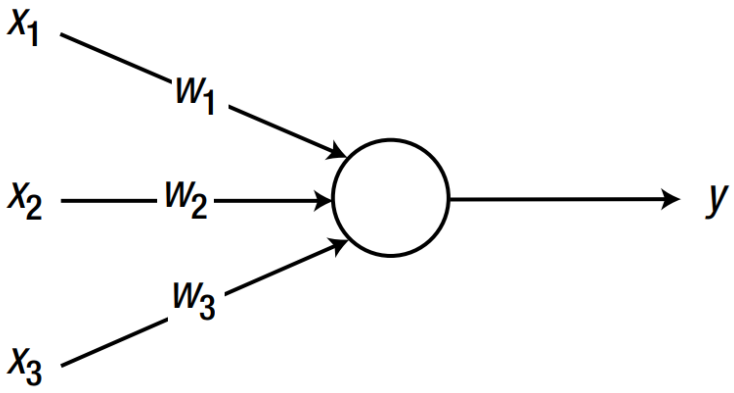

Figure 2: Our same neural network

3. Assume that we have a new correct output, $[0, 1, 1, 0]$. It is different from that of previous one, and we use this new correct output to run Program 3(i.e., testDeltaSGD.m)

Program 9: testDeltaSGDXOR

Listing 2: testDeltaSGDXOR.m

```
clear

data_input = [ 0, 0, 1; 0, 1, 1; 1, 0, 1; 1, 1, 1]; % training data
correct_output = [0; 1; 1; 0]; % correct outputs(i.e., labels), note that it
    is different from previous [0; 0; 1; 1]
weight = 2 * rand(1, 3) - 1; % initializes the weights  with random real
   numbers between [-1, 1]

for epoch = 1 : 40000
    weight =  DeltaSGD(weight, data_input, correct_output)
end

% inference
N = 4;
for k = 1:N
    x = data_input(k, :)';
    v = weight * x;
    y = sigmoid(v)
end
```

Output 9:

```
...
weight =
   -0.2375   -0.1188    0.1188
weight =
   -0.2375   -0.1188    0.1188
y =
    0.5297
y =
    0.5000
y =
    0.4703
y =
    0.4409
```

4. The output is very strange, and training the neural network for a longer period does not make a difference.Why?

$$\begin{bmatrix} 0.5297 \\ 0.5000 \\ 0.4703 \\ 0.4409 \end{bmatrix} \Leftrightarrow D = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Figure 3: The output is very strange

5. In order to answer this question, we'd better visualize the training data. The new training data that testDeltaSGDXOR.m use can be visualized as blew

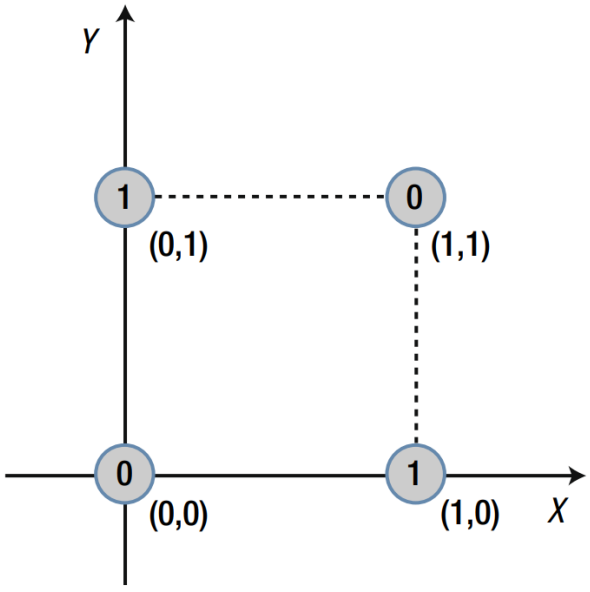

Figure 4: Interpreting the three values of the input data as the X, Y, and Z coordinates

6. One thing to notice from this figure is that we cannot divide the regions of 0 and 1 with a straight line. However, we may divide it with a complicated curve, as shown in Figure 5. This type of problem is said to be **linearly inseparable**.

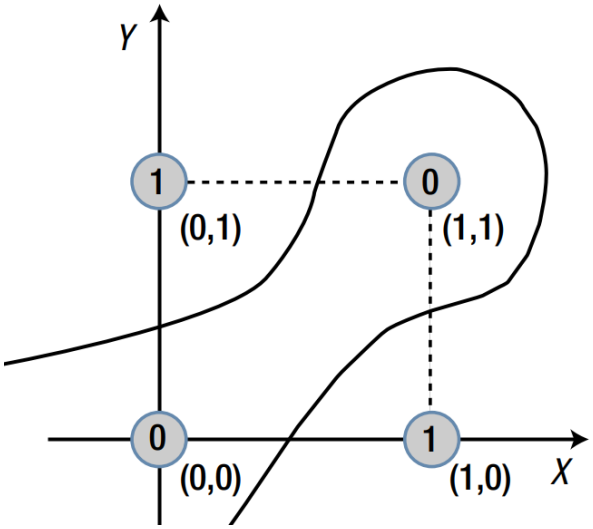

Figure 5: Linearly inseparable

7. But if we visualize the previous training data(e.g., the data that Note II and Note III use), we can find a straight border line that divides the regions of 0 and 1 can be found easily. This is a **linearly separable** problem. In sum, **the single-layer neural network can only solve linearly separable problems**.

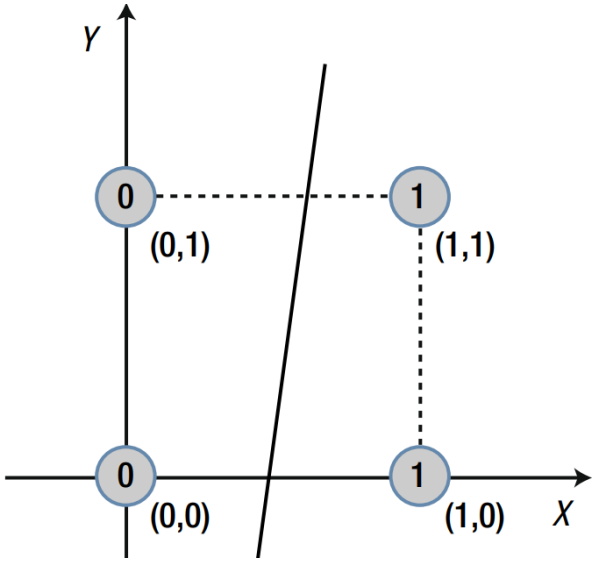

Figure 6: Linearly separable