**Assignment 2**

**KIRAN VENKATESH KULKARNI**

**ID: 1001848434**

* For this assignment I have used python to construct the neural network

Instructions to run the notebook:

* In order to run the notebook please install (numpy, keras, pickle)

**Designing the neural network:**

* I have created a base class to implement forward and backward functions:

class Layer:

def \_\_init\_\_(self,input\_size,output\_size):

self.input\_size=input\_size

self.output\_size=output\_size

def forward(self,input):

pass

def backward(self,output\_error,learning\_size):

pass

* Inheriting the layer class I have created classes for Linear, Sigmoid function, Hyperbolic Tangent function, Softmax function, Negative Log Likelihood function all of which implement both forward and backward functions.
* I have also provided the option to save the weights of the model

**Solving the XOR Problem:**

* The XOR problem is straightforward, with the inputs as [0,0], [0,1], [1,0], [1,1] the model should provide the output values close to [0], [1], [1], [0].
* With 100 epochs and a learning rate of 0.1, I created a neural network as follows:

|  |  |  |
| --- | --- | --- |
| Input | Predicted | Actual |
| [0,0] | [0] | [0] |
| [0,1] | [0.90] | [1] |
| [1,0] | [0.884] | [1] |
| [1,1] | [0] | [0] |

* As we can see the model provided great results. The weights are saved as XOR\_s.w.npy.

**Handwritten Digit Recognition:**

For this part we have selected the MNIST dataset. We experiment with hyperparameters:

Case 1:

* 2 hidden layers
* 5 epochs
* 0.1 learning rate
* Accuracy 0.775
* MSE 0.03264170696928627
* 1/5, error=0.261487
* 2/5, error=0.132127
* 3/5, error=0.088347
* 4/5, error=0.070426
* 5/5, error=0.060365

This resulted in 78% accuracy.

Case 2:

* 1 hidden layer
* 100 epochs
* 0.01 learning rate

Accuracy: 0.80

MSE: 0.029

1/100,error=0.803280

2/100,error=0.548601

3/100,error=0.450870

4/100,error=0.381073

5/100,error=0.329501

6/100,error=0.290046

7/100,error=0.259214

8/100,error=0.234753

9/100,error=0.215052

10/100,error=0.198920

11/100,error=0.185458

12/100,error=0.174010

13/100,error=0.164118

14/100,error=0.155458

15/100,error=0.147793

16/100,error=0.140945

17/100,error=0.134777

18/100,error=0.129183

19/100,error=0.124081

20/100,error=0.119403

21/100,error=0.115097

22/100,error=0.111117

23/100,error=0.107426

24/100,error=0.103993

25/100,error=0.100792

26/100,error=0.097798

27/100,error=0.094992

28/100,error=0.092356

29/100,error=0.089875

30/100,error=0.087534

31/100,error=0.085321

32/100,error=0.083225

33/100,error=0.081236

34/100,error=0.079346

35/100,error=0.077546

36/100,error=0.075830

37/100,error=0.074191

38/100,error=0.072624

39/100,error=0.071124

40/100,error=0.069685

41/100,error=0.068304

42/100,error=0.066976

43/100,error=0.065699

44/100,error=0.064468

45/100,error=0.063281

46/100,error=0.062135

47/100,error=0.061028

48/100,error=0.059957

49/100,error=0.058920

50/100,error=0.057916

51/100,error=0.056942

52/100,error=0.055998

53/100,error=0.055081

54/100,error=0.054190

55/100,error=0.053324

56/100,error=0.052481

57/100,error=0.051662

58/100,error=0.050864

59/100,error=0.050087

60/100,error=0.049330

61/100,error=0.048592

62/100,error=0.047872

63/100,error=0.047171

64/100,error=0.046486

65/100,error=0.045818

66/100,error=0.045165

67/100,error=0.044528

68/100,error=0.043906

69/100,error=0.043298

70/100,error=0.042704

71/100,error=0.042124

72/100,error=0.041556

73/100,error=0.041001

74/100,error=0.040458

75/100,error=0.039927

76/100,error=0.039408

77/100,error=0.038899

78/100,error=0.038401

79/100,error=0.037914

80/100,error=0.037437

81/100,error=0.036970

82/100,error=0.036512

83/100,error=0.036064

84/100,error=0.035624

85/100,error=0.035194

86/100,error=0.034772

87/100,error=0.034358

88/100,error=0.033952

89/100,error=0.033554

90/100,error=0.033164

91/100,error=0.032781

92/100,error=0.032405

93/100,error=0.032037

94/100,error=0.031675

95/100,error=0.031320

96/100,error=0.030971

97/100,error=0.030628

98/100,error=0.030292

99/100,error=0.029961

100/100,error=0.029636

Predicted Value:7, True Value:7

Predicted Value:2, True Value:2

Predicted Value:1, True Value:1

Predicted Value:7, True Value:0

Predicted Value:4, True Value:4

Predicted Value:1, True Value:1

Predicted Value:4, True Value:4

Predicted Value:9, True Value:9

Predicted Value:2, True Value:5

Predicted Value:7, True Value:9

This resulted in 81% accuracy.

**Playing with Hyperparameters:**

* Initializing the initial weights to zeroes:

1/20,error=0.332946

2/20,error=0.329397

3/20,error=0.322662

4/20,error=0.304503

5/20,error=0.261571

6/20,error=0.219050

7/20,error=0.176217

8/20,error=0.145646

9/20,error=0.123571

10/20,error=0.105211

11/20,error=0.089465

12/20,error=0.076724

13/20,error=0.066960

14/20,error=0.059534

15/20,error=0.053815

16/20,error=0.049281

17/20,error=0.045560

18/20,error=0.042414

19/20,error=0.039687

20/20,error=0.037269

Accuracy: 0.88

MSE: 0.016

Predicted Value:7, True Value:7

Predicted Value:2, True Value:2

Predicted Value:1, True Value:1

Predicted Value:0, True Value:0

Predicted Value:4, True Value:4

Predicted Value:1, True Value:1

Predicted Value:4, True Value:4

Predicted Value:9, True Value:9

Predicted Value:6, True Value:5

Predicted Value:9, True Value:9

The accuracy was 88% when the weights were initialized to zero.

* Initialize the model with random values between -10 and 10:

1/10,error=0.332972

2/10,error=0.329427

3/10,error=0.322204

4/10,error=0.300393

5/10,error=0.260507

6/10,error=0.222607

7/10,error=0.185947

8/10,error=0.152595

9/10,error=0.124835

10/10,error=0.102718

Accuracy 0.8314

MSE 0.027641660830204143

Predicted Value:7, True Value:7

Predicted Value:2, True Value:2

Predicted Value:1, True Value:1

Predicted Value:0, True Value:0

Predicted Value:4, True Value:4

Predicted Value:1, True Value:1

Predicted Value:4, True Value:4

Predicted Value:9, True Value:9

Predicted Value:6, True Value:5

Predicted Value:9, True Value:9

The accuracy was 83% when weights were initialized to values between -10 and 10.

* Training on MNIST dataset with learning rate as 1:

1/20,error=0.277804

2/20,error=0.121712

3/20,error=0.067675

4/20,error=0.049538

5/20,error=0.035140

6/20,error=0.023774

7/20,error=0.017743

8/20,error=0.011884

9/20,error=0.008835

10/20,error=0.006263

11/20,error=0.004692

12/20,error=0.003899

13/20,error=0.003215

14/20,error=0.002719

15/20,error=0.002395

16/20,error=0.002022

17/20,error=0.001701

18/20,error=0.001411

19/20,error=0.001219

20/20,error=0.001049

Accuracy 0.901

MSE 0.015639931861205502

Predicted Value:7, True Value:7

Predicted Value:2, True Value:2

Predicted Value:1, True Value:1

Predicted Value:0, True Value:0

Predicted Value:4, True Value:4

Predicted Value:1, True Value:1

Predicted Value:4, True Value:4

Predicted Value:9, True Value:9

Predicted Value:2, True Value:5

Predicted Value:9, True Value:9

The accuracy was 90% when learning rate was 1.

**Observations:**

* The optimal learning rate was found to be 1 and not 0.001.
* The ideal model would have the hyperparameters : learning rate - 1, epochs - 20, and 1 hidden layer of n nodes.