VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF SCIENCE
COMPUTER VISION

DIGITAL IMAGE & VIDEO PROCESSING

LAB 02 REPORT

Teacher: Ly Quoc Ngoc
Nguyen Manh Hung

Student: Duong Minh Anh Khoi – 20127212

Ho Chi Minh city - 2022

Contents

1 Function definition sigmoid and corresponding derivative

```
21
22 # sigmoid activation
23 def sigmoid(s):
24     return (torch.exp(s) - torch.exp(-s)) / (torch.exp(s) + torch.exp(-s))
25
26 # derivative of sigmoid
27 def sigmoid_derivative(s):
28     return (4 * torch.exp(2*s)) / ((1 + torch.exp(2*s)) * (1 + torch.exp(2*s)))
29
```

2 PyTorch Neural Network

- Initialization function

```
30 # Feed Forward Neural Network Class
31 class FFNN(nn.Module):
32     # initialization function
33     def __init__(self):
34         # init function of base class
35         super(FFNN, self).__init__()
36         # corresponding size of each layer
37         self.inputSize = 4
38         self.hiddenSize = 4
39         self.outputSize = 1
```

- Input, output value for training and input x for predicting

```
113
114 # sample input and output value for training
115 X = torch.tensor(([2, 9, 0, 6], [1, 5, 1, 1], [3, 6, 2, 4], [2, 1, 4, 5]), dtype=torch.float)  # 4 X 4 tensor
116 y = torch.tensor(([90], [100], [88], [70]), dtype=torch.float)  # 4 X 1 tensor
117
118 # scale units by max value
119 X_max, _ = torch.max(X, 0)
120 X = torch.div(X, X_max)
121 y = y / 100  # for max test score is 100
122
123 # sample input x for predicting
124 x_predict = torch.tensor(([3, 8, 4, 2]), dtype=torch.float)  # 1 X 4 tensor
125
```

- Process speed

```
# trains the NN 1,000 times
for i in range(1000):
    # print mean sum squared loss
    print("#" + str(i) + " Loss: " + str(torch.mean((y - NN(X)) ** 2).detach().item()))
    # training with learning rate = 0.4
    NN.train(X, y, 0.4)
```

3 Result

```
#987 Loss: 0.0012335798237472773
#988 Loss: 0.0008912755292840302
#989 Loss: 0.001230903435498476
#990 Loss: 0.0008896234212443233
#991 Loss: 0.0012282377574592829
#992 Loss: 0.0008879758534021676
#993 Loss: 0.0012255802284926176
#994 Loss: 0.0008863348048180342
#995 Loss: 0.0012229431886225939
#996 Loss: 0.0008847099961712956
#997 Loss: 0.0012203154619783163
#998 Loss: 0.0008830741862766445
#999 Loss: 0.0012176738819107413
Predicted data based on trained weights:
Input:
tensor([0.3750, 1.0000, 0.5000, 0.2500])
Output:
tensor([0.9982])
```