**Solutions to hw1**

**Xian Sun**

Problem1

Q 1.1

True.

From characteristics of convex function, we know that local minimum is also global minimum. Thus, if we use proper learning rate and initialization, gradient descent will reach closer and closer to a minimum until it reaches the minimum.

Q 1.2

False.

Too many layers may lead to overfitting problem.

Q 1.3

False.

Different set of weights may perfectly lead to 100% classification accuracy. Therefore, the final set of weights may be different if the Madaline is initialized with different values.

Q 1.4

False.

It is in the sides of curve not in the middle, where gradient becomes zero. Besides, for a small number of layers, Sigmoid neurons are not likely to die during the training process and they work very well. But for large number of layers, we do not use Sigmoid neurons usually because it is more likely to “die”.

Q 1.5

False.

Increasing the height and width of the input feature map will not always lead to a larger output feature map size. For instace, if the kernel size is 5x5, stride is 2, no padding, if the original size of input is 10x10. If we input a larger input like 11x11, the output size will not change.

Problem2

Q2.1

The logic function is AND

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | s | y |
| -1 | -1 | -3.5 | -1 |
| -1 | +1 | -1.5 | -1 |
| +1 | -1 | -1.5 | -1 |
| +1 | +1 | 0.5 | +1 |

Q2.2

w0 = -1, w1 = w2 = -1

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | s | y |
| -1 | -1 | 1 | +1 |
| -1 | +1 | -1 | -1 |
| +1 | -1 | -1 | -1 |
| +1 | +1 | -3 | -1 |

Q2.3

w0 = 0, w1 = w2 = w3 = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X1 | X2 | X3 | s | y |
| -1 | -1 | -1 | -3 | -1 |
| -1 | -1 | +1 | -1 | -1 |
| -1 | +1 | -1 | -1 | -1 |
| -1 | +1 | +1 | 1 | +1 |
| +1 | -1 | -1 | -1 | -1 |
| +1 | -1 | +1 | 1 | +1 |
| +1 | +1 | -1 | 1 | +1 |
| +1 | +1 | +1 | 3 | +1 |

Q2.4

w21 = w22 = 1, w20 = 1.5

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | s | y |
| -1 | -1 | -0.5 | -1 |
| -1 | +1 | 1.5 | +1 |
| +1 | -1 | 1.5 | +1 |
| +1 | +1 | -0.5 | -1 |

Problem3

Q 3.1

Q 3.2

Problem4

4.1

4.2

From the hint in the problem, I think it is a “sharping” kernel which emphasizes differences in adjacent pixel values. It will make image (input) more vivid. Besides, we can know from the result in 4.1, the difference between adjacent pixels along the edge are larger than before.

Lab1

1. Run LMS for 20 epochs with learning rate r = 0.01, the weight and MSE loss figure is blelow.

A picture containing game

Description automatically generated

1. From the 3D figure below, we can find out Both of two linear models fit well. The gray plane is a model in a and the pink plane is b model.

A close up of a map

Description automatically generated

1. Plot MSE losses with 4 sets of experiments below.

A picture containing colorful, display, air

Description automatically generated

Set r = 1 and plot the MSE loss below.

**A close up of smoke

Description automatically generated**

Quality of the learning process: From the figure above, if the learning rate is very large like 1, it may be impossible for us to get the optimum as it diverges, which will have a bad impact on our training process. If r = 1, the model will diverge from the optimum and reach the infinity. If r = 0.5, we cannot get the optimum either.

Speed of the learning process: from figures above, if the learning rate is not too large to get an optimum, the smaller learning rate is, less times of epochs we need to reach the optimum. In other words, the smaller learning rate we used, the more epochs it takes.

LAB2

A screenshot of a cell phone

Description automatically generated

# Create the neural network module: LeNet-5

class LeNet5(nn.Module):

def \_\_init\_\_(self):

super(LeNet5, self).\_\_init\_\_()

# Layer definition

self.conv1 = CONV(3,6,5) #Your code here

self.conv2 = CONV(6,16,5) #Your code here

self.fc1 = FC(400, 120) #Your code here

self.fc2 = FC(120, 84) #Your code here

self.fc3 = FC(84, 10) #Your code here

def forward(self, x):

# Forward pass computation

# Conv 1

out = F.relu(self.conv1(x))

# MaxPool

mp1 = nn.MaxPool2d(2,2)

out = mp1(out)

# Conv 2

out = F.relu(self.conv2(out))

# MaxPool

mp2 = nn.MaxPool2d(2,2)

out = mp2(out)

# Flatten

out = out.reshape(out.shape[0],-1)

# FC 1

out = F.relu(self.fc1(out))

# FC 2

out = F.relu(self.fc2(out))

# FC 3

out = self.fc3(out)

return out

# GPU check

device = 'cuda' if torch.cuda.is\_available() else 'cpu'

if device =='cuda':

print("Run on GPU...")

else:

print("Run on CPU...")

# Model Definition

net = LeNet5()

net = net.to(device)

# Test forward pass

data = torch.randn(5,3,32,32)

data = data.to(device)

# Forward pass "data" through "net" to get output "out"

out = net(data) #Your code here

# Check output shape

assert(out.detach().cpu().numpy().shape == (5,10))

print("Forward pass successful")

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Layer | Input shape | Output shape | Weight shape | # Param | # MAC |
| Conv 1 | (1, 3, 32, 32) | (1, 6, 28, 28) | (6, 3, 5, 5) | 450 | 352800.0 |
| Conv 2 | (1, 6, 14, 14) | (1, 16, 10, 10) | (16, 6, 5, 5) | 2400 | 240000.0 |
| FC1 | (1, 400) | (1, 120) | (120, 400) | 48000 | 48000.0 |
| FC2 | (1, 120) | (1, 84) | (84, 120) | 10080 | 10080.0 |
| FC3 | (1, 84) | (1, 10) | (10, 84) | 840 | 840.0 |

1. Results to shape size

Table2

LAB3

A picture containing drawing

Description automatically generatedA picture containing drawing

Description automatically generated

A picture containing drawing

Description automatically generatedA picture containing drawing

Description automatically generated

A picture containing drawing

Description automatically generated

A picture containing drawing

Description automatically generatedA picture containing drawing

Description automatically generated

A picture containing drawing

Description automatically generatedA picture containing drawing

Description automatically generated

A screenshot of a cell phone

Description automatically generated

1. Comparison: Histograms in (c), almost all gradients in other layers are zeros while we have all kinds of values of gradients in (b).

Analysis: This is because we set all weights to 0 in c(c). The process of backpass is chain rule and once the weights are zeros, it will pass zeros to each layer one by one. It is not good for us to set all weights to zeros in CNN model as the results in (c) show because our training process may be dead forever.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A close up of a logo

Description automatically generated