

1. [활성함수] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (4점)

(코드의 해석과 결과의 의미를 작성하세요.)

```
# 파이썬 2, 파이썬 3 지원
from __future__ import division, print_function, unicode_literals

# 관련 라이브러리
import os
import numpy as np
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

def logit(z):
    return 1 / (1 + np.exp(-z))

def relu(z):
    return np.maximum(0, z)

def derivative(f, z, eps=0.000001):
    return (f(z + eps) - f(z - eps))/(2 * eps)

z = np.linspace(-5, 5, 200)

plt.figure(figsize=(11,4))

plt.subplot(121)
plt.plot(z, np.sign(z), "r-", linewidth=2, label="step")
plt.plot(z, logit(z), "g--", linewidth=2, label="sigmoid")
plt.plot(z, np.tanh(z), "b-", linewidth=2, label="tanh")
plt.plot(z, relu(z), "m-.", linewidth=2, label="ReLU")
plt.grid(True)
plt.legend(loc="center right", fontsize=14)
plt.title("activation function: g(z)", fontsize=14)
plt.axis([-5, 5, -1.2, 1.2])

plt.subplot(122)
plt.plot(z, derivative(np.sign, z), "r-", linewidth=2, label="step ")
plt.plot(0, 0, "ro", markersize=5)
```

```

plt.plot(0, 0, "rx", markersize=10)
plt.plot(z, derivative(logit, z), "g--", linewidth=2, label="sigmoid")
plt.plot(z, derivative(np.tanh, z), "b-", linewidth=2, label="tanh")
plt.plot(z, derivative(relu, z), "m-.", linewidth=2, label="ReLU")
plt.grid(True)
plt.title("gradient: g'(z)", fontsize=14)
plt.axis([-5, 5, -0.2, 1.2])

plt.show()

```

(1) 화면 출력 확인 및 각 활성함수의 특징을 비교 서술

2. [오류 역전파] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (4점)

(코드의 해석과 결과의 의미를 작성하세요.)

```

import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

```

(1) 해당 연산망의 그래프 연산을 손으로 작성

```

grad_c = 1.0
grad_b = grad_c * np.ones ((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a*y
grad_y = grad_a*x

```

(2) grad_c, grad_b, grad_a, grad_z, grad_x, grad_y 출력 확인

```
import torch
x = torch.randn(N, D, requires_grad=True)
y = torch.randn(N, D, requires_grad=True)
z = torch.randn(N, D)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()
# (3) grad_x, grad_y 출력 확인
```

3. [오류 역전파] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (4점)

(코드의 해석과 결과의 의미를 작성하세요.)

```
import torch

x = torch.randn(1, 10)
prev_h = torch.randn(1, 20)
w_h = torch.randn(20, 20)
w_x = torch.randn(20, 10)

i2h = torch.mm(w_x, x.t())
h2h = torch.mm(w_h, prev_h.t())
next_h = i2h + h2h
next_h = next_h.tanh()
# (1) 해당 신경망의 그래프 연산을 손으로 작성
```

```
loss = next_h.sum()
loss.backward()
# (2) loss 출력 확인
```

4. [신경망 학습] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (4점)

(코드의 해석과 결과의 의미를 작성하세요.)

```
import torch

N, D_in, H, D_out = 64, 1000, 100, 10

x = torch.randn(N, D_in)
y = torch.randn(D, D_out)
w1 = torch.randn(D_in, H)
w2 = torch.randn(H, D_out)

learning_rate = 10e-6
for t in range(500):
    h = x.mm(w1)
    h_relu = h.clamp(min=0)
    y_pred = h_relu.mm(w2)
    loss = (y_pred - y).pow(2).sum()

    grad_y_pred = 2.0 * (y_pred - y)
    grad_w2 = h_relu.t().mm(grad_y_pred)
    grad_h_relu = grad_y_pred.mm(w2.t())
    grad_h = grad_relu.clone()
    grad_h[h < 0] = 0
    grad_w1 = x.t().mm(grad_h)

    w1 -= learning_rate * grad_w1
    w2 -= learning_rate * grad_w2
```

(1) 매 t마다 y_pred, loss 변화를 화면 출력 확인 (plot)

(2) 해당 학습이 적절히 진행되고 있는지 서술

5. [신경망 학습] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (4점)

(코드의 해석과 결과의 의미를 작성하세요.)

```

import torch
N, D_in, H, D_out = 64, 1000, 100, 10

x = torch.randn(N, D_in)
y = torch.randn(D, D_out)
w1 = torch.randn(D_in, H, requires_grad=True)
w2 = torch.randn(H, D_out, requires_grad=True)

learning_rate = 10e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    loss.backward()

    with torch.no_grad():
        w1 -= learning_rate * w1.grad
        w2 -= learning_rate * w2.grad
        w1.grad.zero_()
        w2.grad.zero_()

```

(1) 매 t마다 y_pred, loss 변화를 화면 출력 확인 (plot)

(2) 앞 문제의 코드와 비교

6. [신경망 학습] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (4점)

(코드의 해석과 결과의 의미를 작성하세요.)

```

import torch

class MyReLU(torch.autograd.Function):
    @staticmethod
    def forward(ctx, x):
        ctx.save_for_backward(x)
        return x.clamp(min=0)

    @staticmethod

```

```

def backward(ctx, grad_y):
    x, = ctx.saved_tensors
    grad_input = grad_y.clone()
    grad_input[x < 0] = 0
    return grad_input

def my_relu(x):
    return MyReLU.apply(x)

N, D_in, H, D_out = 64, 1000, 100, 10

x = torch.randn(N, D_in)
y = torch.randn(D, D_out)
w1 = torch.randn(D_in, H, requires_grad=True)
w2 = torch.randn(H, D_out, requires_grad=True)

learning_rate = 10e-6
for t in range(500):
    y_pred = my_relu(x.mm(w1)).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    loss.backward()

    with torch.no_grad():
        w1 -= learning_rate * w1.grad
        w2 -= learning_rate * w2.grad
        w1.grad.zero_()
        w2.grad.zero_()

# (1) 매 t마다 y_pred, loss 변화를 화면 출력 확인 (plot)
# (2) 앞 문제의 코드와 비교

```

7. [신경망 학습] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (4점)

(코드의 해석과 결과의 의미를 작성하세요.)

```

import torch

class TwoLayerNet(torch.nn.Module):
    def __init__(self, D_in, H, D_out):
        super(TwoLayerNet, self).__init__()
        self.linear1 = torch.nn.Linear(D_in, H)
        self.linear2 = torch.nn.Linear(H, D_out)

    def forward(self, x):
        h_relu = self.linear1(x).clamp(min=0)
        y_pred = self.linear2(h_relu)
        return y_pred

N, D_in, H, D_out = 64, 1000, 100, 10

x = torch.randn(N, D_in)
y = torch.randn(D, D_out)

model = TwoLayerNet(D_in, H, D_out)

optimizer = torch.optim.SGD(model.parameters(), lr=1e-4)

for t in range(500):
    y_pred = model(x)
    loss = torch.nn.functional.mse(y_pred, y)

    loss.backward()
    optimizer.step()
    optimizer.zero_grad()

# (1) 매 t마다 y_pred, loss 변화를 화면 출력 확인 (plot)
# (2) 앞 문제의 코드와 비교

```

8. [데이터 전처리] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (4점)

(코드의 해석과 결과의 의미를 작성하세요.)

```
import torch
from torch.utils.data import TensorDataset, DataLoader

N, D_in, H, D_out = 64, 1000, 100, 10

x = torch.randn(N, D_in)
y = torch.randn(D, D_out)

loader = DataLoader(TensorDataset(x, y), batch_size=8)
model = TwoLayerNet(D_in, H, D_out)

optimizer = torch.optim.SGD(model.parameters(), lr=1e-2)

for epoch in range(20):
    for x_batch, y_batch in loader:
        y_pred = model(x_batch)
        loss = torch.nn.functional.mse(y_pred, y_batch)

        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
```

(1) 매 세대^{epoch}마다 y_pred, loss 변화를 화면 출력 확인 (plot)

(2) 앞 문제의 코드와 비교

9. [영상 인식] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (12점)

(코드의 해석과 결과의 의미를 작성하세요.)

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
```

```
from torch.autograd import Variable
import matplotlib.pyplot as plt
%matplotlib inline

is_cuda=False
if torch.cuda.is_available():
    is_cuda = True

transformation = transforms.Compose([transforms.ToTensor(),transforms.Normalize((0.1307,), (0.3081,))])

train_dataset = datasets.MNIST('data/',train=True,transform=transformation,download=True)
test_dataset = datasets.MNIST('data/',train=False,transform=transformation,download=True)

train_loader = torch.utils.data.DataLoader(train_dataset,batch_size=32,shuffle=True)
test_loader = torch.utils.data.DataLoader(test_dataset,batch_size=32,shuffle=True)

sample_data = next(iter(train_loader))

def plot_img(image):
    image = image.numpy()[0]
    mean = 0.1307
    std = 0.3081
    image = ((mean * image) + std)
    plt.imshow(image,cmap='gray')

plot_img(sample_data[0][2])
# (1) 화면 출력 확인

plot_img(sample_data[0][1])
# (2) 화면 출력 확인

class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)
```

```

def forward(self, x):
    x = F.relu(F.max_pool2d(self.conv1(x), 2))
    x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
    x = x.view(-1, 320)
    x = F.relu(self.fc1(x))
    #x = F.dropout(x,p=0.1, training=self.training)
    x = self.fc2(x)
    return F.log_softmax(x,dim=1)

model = Net()
if is_cuda:
    model.cuda()

optimizer = optim.SGD(model.parameters(),lr=0.01)

data , target = next(iter(train_loader))

output = model(Variable(data.cuda()))

# (3) output.size() 출력 확인

# (4) target.size() 출력 확인

def fit(epoch,model,data_loader,phase='training',volatile=False):
    if phase == 'training':
        model.train()
    if phase == 'validation':
        model.eval()
        volatile=True
    running_loss = 0.0
    running_correct = 0
    for batch_idx , (data,target) in enumerate(data_loader):
        if is_cuda:
            data,target = data.cuda(),target.cuda()
        data , target = Variable(data,volatile),Variable(target)
        if phase == 'training':
            optimizer.zero_grad()
        output = model(data)

```

```

loss = F.nll_loss(output,target)

running_loss += F.nll_loss(output,target,size_average=False).data[0]
preds = output.data.max(dim=1,keepdim=True)[1]
running_correct += preds.eq(target.data.view_as(preds)).cpu().sum()

if phase == 'training':
    loss.backward()
    optimizer.step()

loss = running_loss/len(data_loader.dataset)
accuracy = 100. * running_correct/len(data_loader.dataset)

print(f'{phase} loss is {loss:.2f} and {phase} accuracy is {(running_correct)/len(data_loader.dataset)}{accuracy:.4f}')
return loss, accuracy

train_losses , train_accuracy = [],[]
val_losses , val_accuracy = [],[]
for epoch in range(1,20):
    epoch_loss, epoch_accuracy = fit(epoch,model,train_loader,phase='training')
    val_epoch_loss , val_epoch_accuracy = fit(epoch,model,test_loader,phase='validation')
    train_losses.append(epoch_loss)
    train_accuracy.append(epoch_accuracy)
    val_losses.append(val_epoch_loss)
    val_accuracy.append(val_epoch_accuracy)

# (5) 화면 출력 확인

plt.plot(range(1,len(train_losses)+1),train_losses,'bo',label = 'training loss')
plt.plot(range(1,len(val_losses)+1),val_losses,'r',label = 'validation loss')
plt.legend()

# (6) 화면 출력 확인

plt.plot(range(1,len(train_accuracy)+1),train_accuracy,'bo',label = 'train accuracy')
plt.plot(range(1,len(val_accuracy)+1),val_accuracy,'r',label = 'val accuracy')
plt.legend()

# (7) 화면 출력 확인

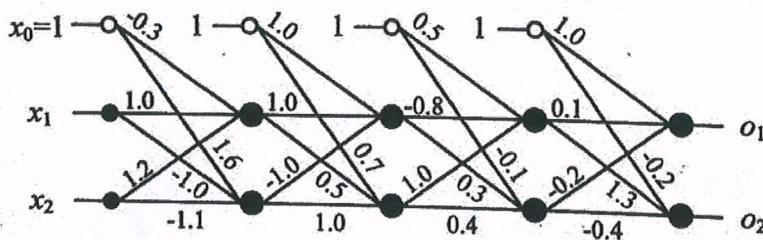
```

10. NOR 게이트와 AND 게이트의 동작을 데이터로 간주하면 다음과 같다. 이들을 100% 옳게 분류하는 퍼셉트론을 각각 제시하시오.

$$\text{NOR 분류} \quad \begin{cases} \mathbf{x}_1 = (0,0)^T, y_1 = 1 \\ \mathbf{x}_2 = (1,0)^T, y_2 = -1 \\ \mathbf{x}_3 = (0,1)^T, y_3 = -1 \\ \mathbf{x}_4 = (1,1)^T, y_4 = -1 \end{cases} \quad \text{AND 분류} \quad \begin{cases} \mathbf{x}_1 = (0,0)^T, y_1 = -1 \\ \mathbf{x}_2 = (1,0)^T, y_2 = -1 \\ \mathbf{x}_3 = (0,1)^T, y_3 = -1 \\ \mathbf{x}_4 = (1,1)^T, y_4 = 1 \end{cases}$$

11. 다음은 은닉층이 3개인 DMLP이다.

Hint 계산은 Matlab 또는 Python을 사용하시오.



- (1) 가중치 행렬 $\mathbf{U}^1, \mathbf{U}^2, \mathbf{U}^3, \mathbf{U}^4$ 를 식 (4.1)처럼 쓰시오.
- (2) $\mathbf{x} = (1,0)^T$ 가 입력되었을 때 출력 \mathbf{o} 를 구하시오. 활성함수로 로지스틱 시그모이드를 사용하시오.
- (3) $\mathbf{x} = (1,0)^T$ 가 입력되었을 때 출력 \mathbf{o} 를 구하시오. 활성함수로 ReLU를 사용하시오.
- (4) $\mathbf{x} = (1,0)^T$ 의 기대 출력이 $\mathbf{o} = (0,1)^T$ 일 때, 현재 1.0인 u_{12}^3 가중치를 0.9로 줄이면 오류에 어떤 영향을 미치는지 설명하시오.

12. [그림 4-14]에서 특징 맵의 나머지 8개 값을 계산하세요.

3*3*3 입력 영상

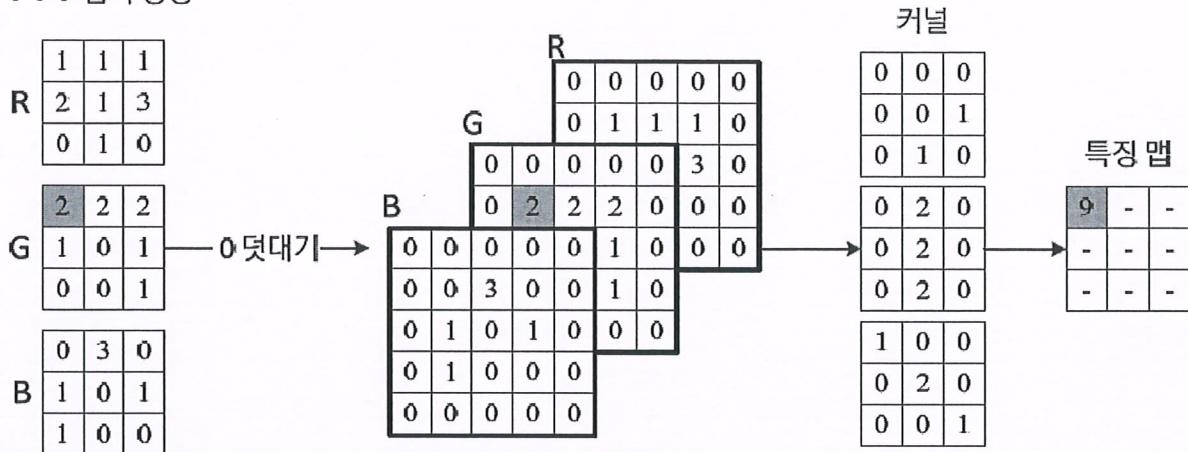
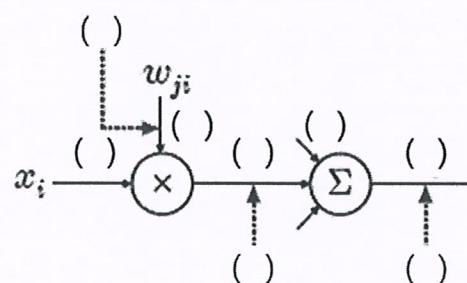


그림 4-14 텐서의 컨볼루션 연산(0 덧대기 적용)

13. 컨볼루션 층의 입력 크기가 $32 \times 32 \times 3$ 이고, (a) 10개 5×5 필터들을 보폭 1과 덧대기 2로 적용하였을 때 출력의 크기와 매개변수의 수를 구하세요. (b) 동일한 입력에 64개 3×3 필터들을 보폭 1과 덧대기 1로 적용하였을 때 출력의 크기와 매개변수의 수도 구하세요. (6점)

14. 아래 그림의 연산 그래프 예처럼 $f(x, y, z) = (x+y)z$ 연산에 대한 연산 그래프를 새롭게 생성하고, $x=-2, y=5, z=-4$ 인 경우에 전방 전파와 이에 대응되는 오류 역전파를 각 가중치마다 계산하세요.

[예에 표시된 것처럼 전방 전파 연산 결과는 검은색 빙간, 오류 역전파 연산 결과는 빨간색 빙간으로 표시하여 구분하세요.]



표시된 점수 외의 문제의 점수는 모두 5점